NATIONAL BESTSELLER
MORE THAN 2,000,000 How It Works SOLD

# How Computers Work

BY RON WHITE

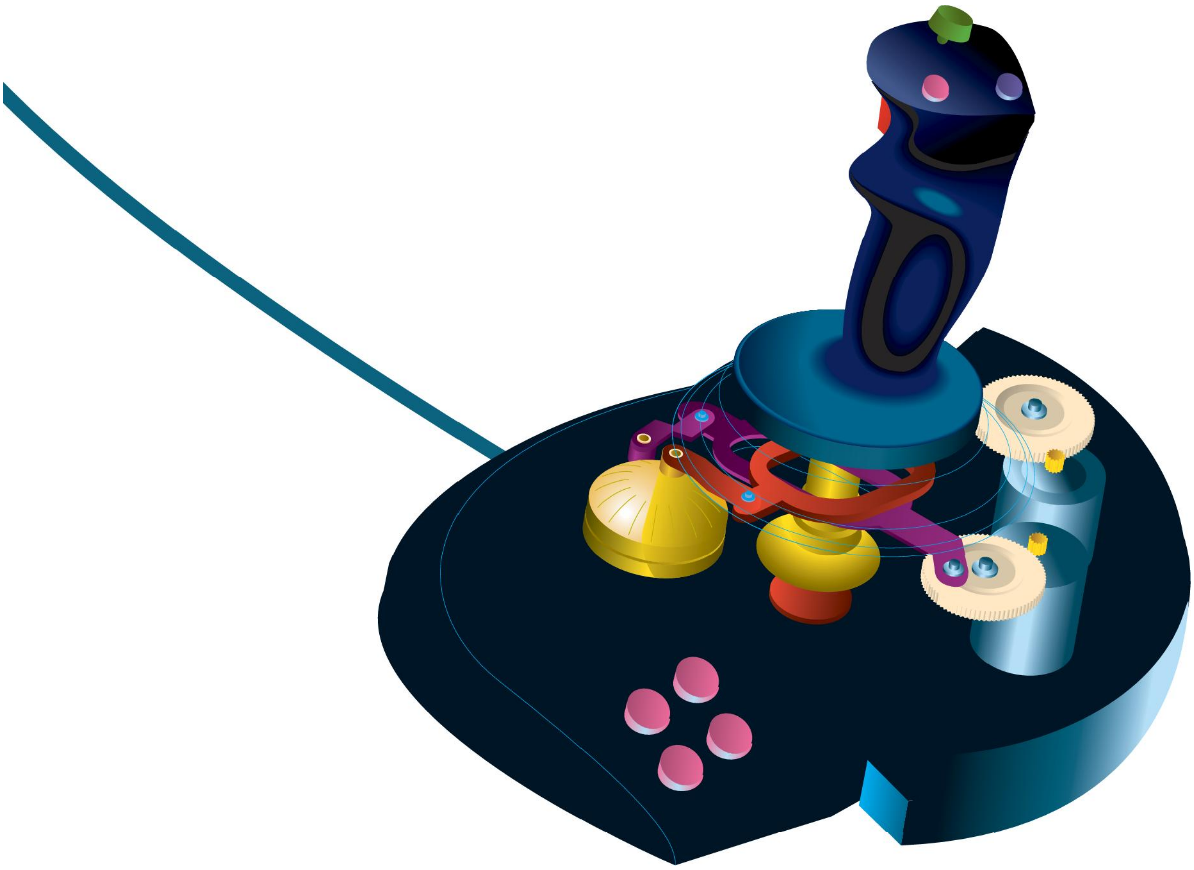
ILLUSTRATED BY TIM DOWNS

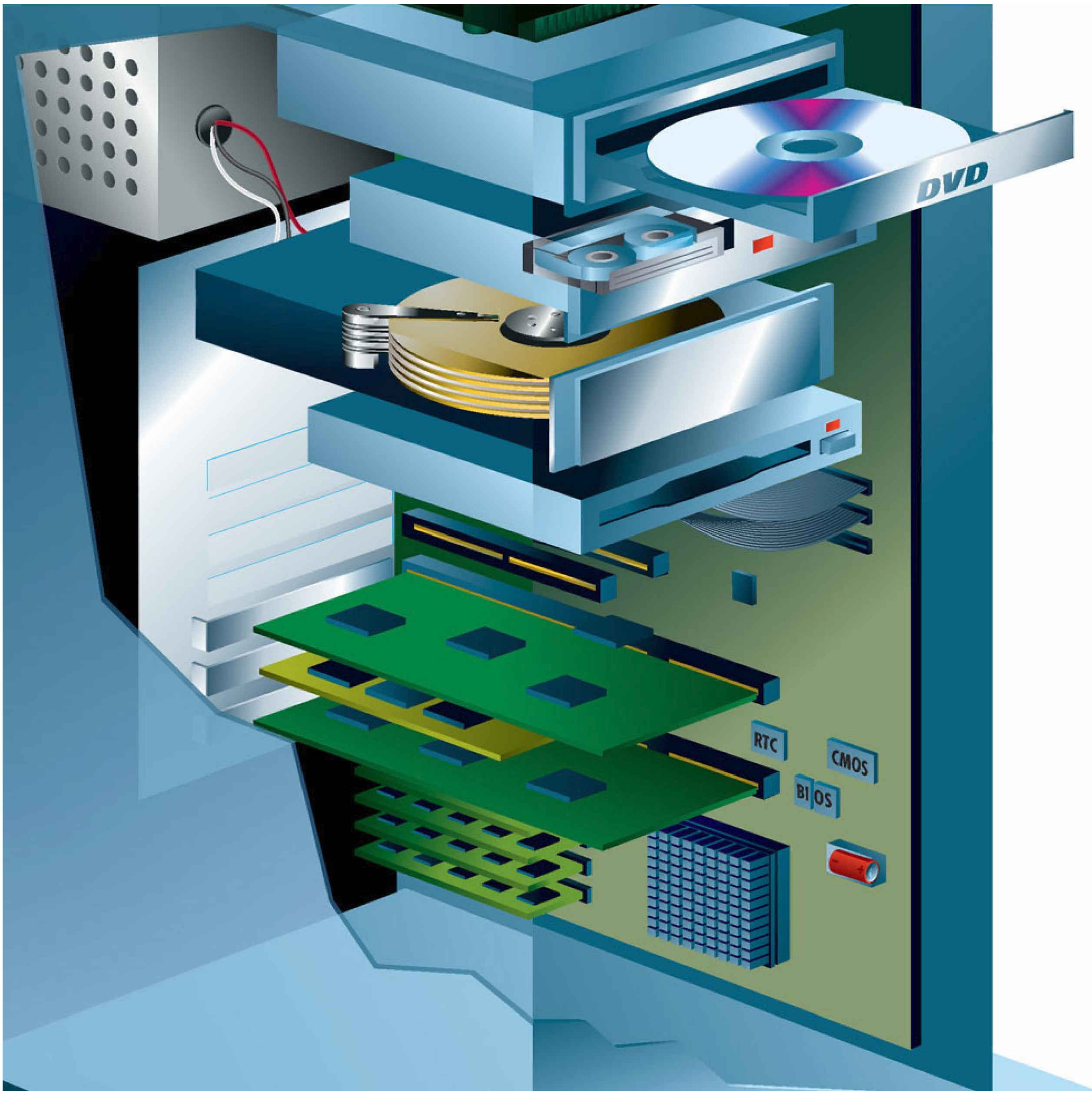
NINTH EDITION

QUE

# How Computers Work

*Ninth Edition*




DVD
RTC
CMOS
BIOS

# How Computers Work

## *Ninth Edition*


## Ron White

**Illustrated by Timothy Edward Downs**



QUE®

800 East 96th Street
Indianapolis, IN 46240

**How Computers Work, Ninth Edition**



ISBN-13: 978-0-789-73613-0

ISBN-10: 0-789-73613-6

Library of Congress Cataloging-in-Publication data is on file.

Printed in the United States of America

First Printing: November 2008

### Trademarks



### Warning and Disclaimer



### Bulk Sales

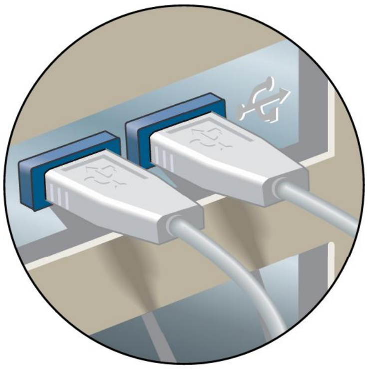
Que Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

**U.S. Corporate and Government Sales**
**1-800-382-3419**
**corpsales@pearsontechgroup.com**

For sales outside of the U.S., please contact

**International Sales**
**international@pearsoned.com**

| | |
|---|---|
| Associate Publisher | Greg Wiegand |
| Acquisitions Editor | Todd Brakke |
| Development Editor | Todd Brakke |
| Managing Editor | Patrick Kanouse |
| Project Editor | Seth Kerney |
| Copy Editor | Megan Shaw |
| Indexer | Ken Johnson |
| Proofreader | Heather Waye Arle |
| Technical Editor | Mark Reddin |
| Publishing Coordinator | Cindy Teeters |
| Book Designer | Anne Jones |



**For Shannon and Michael:**
**Who always kept me honest in my explanations.**
**—Ron**

**For Olivia and Marco, my beloved children. You are the light of my life and my daily inspiration.**
**—Tim**

# About the Author

**RON WHITE** is the former executive editor of *PC Computing* magazine, where he developed the popular *How It Works* illustration to explain the new technologies that were emerging in computing at a prodigious rate. He is also the author of the best-selling *How Digital Photography Works*, and books on software, MP3, and digital cameras. His writing and photography have appeared in some of the leading magazines in the nation. He can be reached at ron@ronwhite.com.

# About the Illustrator

**TIMOTHY EDWARD DOWNS** is the national award-winning illustrator of *How Computers Work* and *How Digital Photography Works*. Tim has been involved in all facets of graphic design in his illustrious career. From illustrator to creative director, Tim has led teams of artists and designers in advertising agencies, marketing communications firms, and consumer magazines to better tell their stories through illustration, photography, typography, and design. "Our job doesn't start when the writer hits Save. In order to effectively communicate the tone or the concept of the piece, we need to know and understand the story from the original brainstorm all the way through final execution," reminds Tim.

Examples of Tim's design, illustration, and photographic work can be seen at http://www.timothyedwarddowns.com.

# Acknowledgments

THE changes in PCs have been so rapid, so extensive, and so complex, I could never have learned the tricks behind the new technologies without the help of many knowledgeable people who were not only generous with that knowledge, but were patient enough to explain it to me until the mental LED lit up.

My first acknowledgment must be to the late Jim Seymour, who was a columnist in *PC Magazine* and *Infoworld*, and who was the editor-in-chief of *PC Computing* when it was launched in 1988. Jim died unexpectedly and left a void in the universe for knowledge, friendship, and serenity. The longer I spend in a world without Jim, the more keenly I feel the loss.

Jim and his wife, Nora, who was editor of *PCC* and a woman of great warmth, charm, and smarts, gave me my first chance at a national magazine. And what a chance it was. I had the privilege of launching the *How It Works* feature in *PC Computing* in 1989, but over the years, many people have worked on the series, and several chapters are based on the research and explanations they've done. Deep thanks go to Margaret Ficklen, Herb Brody, Brett L. Glass, Marty Jerome, Raymond Jones, Matthew Lake, Jack Nimersheim, Randy Ross, Stephen Sagman, Jan Smith, Dylan Tweney, Doug van Kirk, Mark L. Van Name, Bill Catchings, Christine Grech Wendin, and Kenan Woods. I owe a huge debt to Mike Edelhart, who a dozen years ago was editor-in-chief of *PC Computing* and who convinced a dubious publisher to take a chance on a kind of computer book that had never been done before. To Preston Gralla, who guided me in the launch of *How It Works* and who jumped in when I needed help to write several of the spreads, thanks for a friendship and too many acts of kindness to list.

I'm also grateful to the dozens of people in the PC industry who've shared their knowledge, schematics, and white papers to infuse *How Computers Work* with detail and accuracy. I'm particularly grateful to Chris Thornton of ClipMate; Karen Thomas at Olympus; Joe Vanderwater, Bryan Langerdoff, Dan Francisco, Tami Casey, John Hyde, Bill Kircos, Susan Shaw, and Seth Walker at Intel; Russell Sanchez and Adam Kahn of Microsoft; Jim Bartlett, Ellen Reid Smith, Tim Kearns, and Desire Russell at IBM; Barrett Anderson and Todd Hollinshead at id Software; Marcie Pedrazzi of Adaptec; Eileen Algaze at Rockwell; A.J. Rodgers, Jennifer Jones, and Susan Bierma at Tektronix; Ben Yoder and Lisa Santa Anna at Epson; Dewey Hou at TechSmith Corporation; Ray Soneira, author of Sonera Technologies' DisplayMate; Kathryn Brandberry and Doyle Nicholas at Thrustmaster; Dani Johnston at Microtek Lab; Lisa Tillman of The Benjamin Group; Dvorah Samansky at Iomega; Brandon Talaich at SanDisk; Chris Walker with Pioneer USA; Carrie Royce at Cirque Corporation; Andy Marken at Marken Communications; and Tracy Laidlaw at Beyond Words. If there are any mistakes in this book, I can't lay the blame on them.

Thanks to editors now and over the course of nine editions: Rick Kughen, Stephanie McComb, Tonya Simpson, Nick Goetz, Angelina Ward, Sarah Robbins, Juliet Langley, Cindy Hudson, Melinda Levine, Lysa Lewallen, Renee Wilmeth, and Leah Kirkpatrick for pulling everything together. A big debt of gratitude goes to Que Associate Publisher Greg Wiegand for this support for so many years. Thanks to

Mark Reddin and John Rizzo for tech editing, and John for researching several chapters. I'm deeply grateful for the editing of this edition by Todd Brakke, who is the type of intelligent, dedicated editor every writer wants. And thanks to my wife, Sue, for her constant encouragement, spirit-raising, and humor while waiting for me to pop out of writing mode.

I learned long ago that a writer's skill depends largely on how well the writer can steal from others. In addition to just about every book in the Que *How It Works* series, other books were invaluable for details on PC innards: *Understanding I/O Subsystems* by W. David Schwaderer and Andrew W. Wilson, Jr.; *Inside the IBM PC* by Peter Norton; *The PC Configuration Handbook* by John Woram; Marshall Brain's *HowStuffWorks.com; The Winn Rosch Hardware Bible* by Winn L. Rosch; and Scott Mueller's *Upgrading and Repairing PCs*. Also helpful was *The Way Things Work* by David Macaulay, not only for its informative explanations of computers, but for its inspiring examples of how to combine text and art into clear explanations.

Finally, this book would not be what it is without the art work of Timothy Edward Downs. Invariably, he transformed my Neanderthal sketches into clear, informative illustrations, but also managed to make them into wonderful works of art. Tim has created a new, visual way to communicate technology, and I'm proud one of its first appearances was in this book.

# We Want to Hear From You!

AS the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

As an associate publisher for Que Publishing, I welcome your comments. You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books better.

*Please note that I cannot help you with technical problems related to the topic of this book. We do have a User Services group, however, where I will forward specific technical questions related to the book.*

When you write, please be sure to include this book's title and author as well as your name, email address, and phone number. I will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@quepublishing.com

Mail: Greg Wiegand
Associate Publisher
Que Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

# Reader Services

Visit our website and register this book at www.quepublishing.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

# Introduction

*"Any sufficiently advanced technology is indistinguishable from magic."*
***—Arthur C. Clarke***

SORCERERS have their magic wands—powerful, potentially dangerous tools with lives of their own. Witches have their familiars—creatures disguised as household beasts that could, if they choose, wreak the witches' havoc. Mystics have their golems—beings built of wood and tin brought to life to do their masters' bidding.

We have our personal computers.

PCs, too, are powerful creations that often seem to have a life of their own. Usually, they respond to a wave of a mouse or a spoken incantation by performing tasks we couldn't imagine doing ourselves without some sort of preternatural help. But even as computers successfully carry out our commands, it's often difficult to quell the feeling that there's some wizardry at work here.

And then there are the times when our PCs, like malevolent spirits, rebel and open the gates of chaos onto our neatly ordered columns of numbers, our carefully wrought sentences, and our beautifully crafted graphics. When that happens, we're often convinced that we are, indeed, playing with power not entirely under our control. We become sorcerers' apprentices, whose every attempt to right things leads to deeper trouble.

Whether our personal computers are faithful servants or imps, most of us soon realize there's much more going on inside those silent boxes than we really understand. PCs are secretive. Open their tightly sealed cases and you're confronted with poker-faced components. Few give any clues as to what they're about. Most of them consist of sphinx-like microchips that offer no more information about themselves than some obscure code printed on their impenetrable surfaces. The maze of circuit tracings etched on the boards is fascinating, but meaningless, hieroglyphics. Some crucial parts, such as the hard drive and power supply, are sealed with printed omens about the dangers of peeking inside—omens that put to shame the warnings on a pharaoh's tomb.

This book is based on two ideas. One is that the magic we understand is safer and more powerful than the magic we don't. This is not a hands-on how-to book. Don't look for any instructions for taking a screwdriver to this part or the other. But perhaps your knowing more about what's going on inside all those stoic components makes them a little less formidable when something does go awry. The second idea behind this book is that knowledge, in itself, is a worthwhile and enjoyable goal. This book is written to respond to your random musings about the goings-on inside that box you sit in front of several hours a day. If this book puts your questions to rest—or raises new ones—it will have done its job.

At the same time, however, I'm trusting that knowing the secrets behind the magician's legerdemain won't spoil the show. This is a real danger. Mystery is often as compelling as knowledge. I'd hate to think that anything you read in this book takes away that sense of wonder you have when you manage to make your PC do some grand, new trick. I hope that, instead, this book makes you a more confident sorcerer.

# Before You Begin

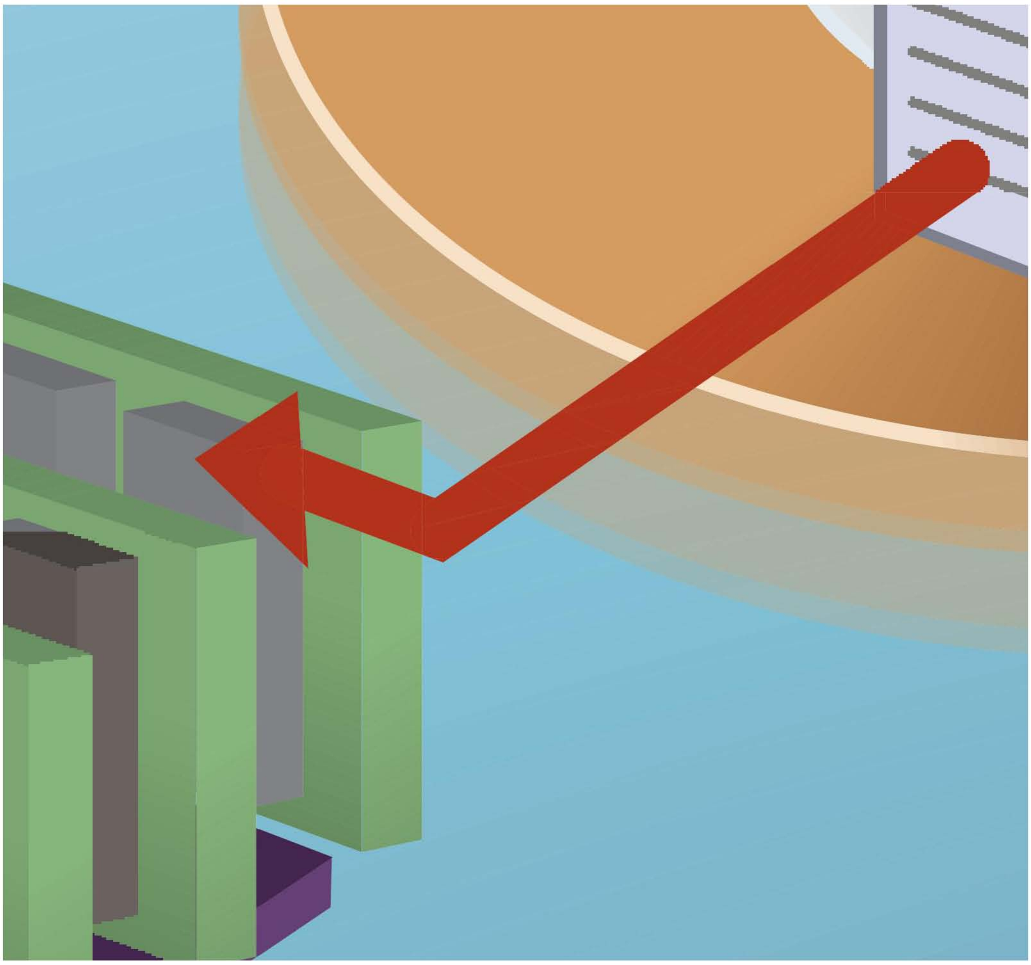
THIS book has been written with a certain type of personal computer in mind—the Wintel, a PC most often built around an Intel processor and running Microsoft Windows. Many of the specifics in these explanations apply only to that class of computer and those components.

In more general terms, the explanations also apply to Macintosh computers, Unix workstations, and even minicomputers and mainframes. But I've made no attempt to devise universal explanations of how computers work. To do so would, of necessity, detract from the understanding that comes from inspecting specific components.

Even so, there is so much variety even within the Intel/Microsoft world of PCs that, at times, I've had to limit my explanations to particular instances or stretch the boundaries of a particular situation to make an explanation as generic as possible. If you spot anything that doesn't seem quite right in this book, I hope that my liberties with the particulars is the only cause.

Ron White

San Antonio, Texas



**30,000 B.C.**
Paleolithic peoples in central Europe record numbers by notching tallies on animal bones, ivory, and stone.

**3400 B.C.**
Egyptians develop a symbol for the number 10, simplifying the representation of large numbers.

**2600 B.C.**
The Chinese introduce the abacus. It was used in China for calculating the census as recently as A.D. 1982.

**300 B.C.**
Euclid's Elements summarizes all the mathematical knowledge of the Greeks. It is used for the next 2,000 years.

**260 B.C.**
The Maya develop a sophisticated base-20 system of mathematics that includes zero.

**1500**
Mechanical calculator invented by Leonardo da Vinci.

**1614**
John Napier describes the nature of logarithms. He also builds Napier's Bones, the forerunner to the slide rule.

**1621**
William Oughtred invents the slide rule, which does not become obsolete for nearly 350 years.

**1642**
Blaise Pascal invents Pascaline, the first mechanical calculator. It was hand turned and could only add and subtract.

**1670**
Gottfried Leibniz improves upon Pascaline by adding multiplication, division, and square root capabilities.

**1679**
Leibniz introduces binary arithmetic.

PART

# 1

# Boot-Up Process

CHAPTERS



**1822**
Charles Babbage invents the Difference Engine, a large mechanical calculator capable of addition and subtraction.

**1830**
Charles Babbage conceives of the Analytical Engine but dies before its completion.

**1890**
Herman Hollerith creates an electric tabulating system for U.S. Census Bureau.

**1896**
Hollerith forms the Tabulating Machine Company, which later becomes International Business Machines.

**1902–1905**
Albert Einstein discovers Theory of Relativity. He publishes it in dissertation at University of Zurich.

**1904**
John Ambrose Fleming develops vacuum tubes.

**1926**
Patent for semiconductor transistor that allowed electrical currents to flow through computer, passing data.

**1936**
Konrad Zuse creates a programmable, digital computing machine that introduces use of binary system and valves.

**1943**
British build Colossus, a machine to break German codes.

**1943–45**
U.S. Army builds ENIAC computer to calculate weapons' trajectories.

*"I think there is a world market for maybe five computers."*

**—Thomas Watson, chairman of IBM, 1943**

**BEFORE** your personal computer is turned on, it is a dead collection of sheet metal, plastic, metallic tracings, and tiny flakes of silicon. When you push the On switch, one little burst of electricity—only about 3–5 volts—starts a string of events that magically brings to life what would otherwise remain an oversized paperweight.

Even with that spark of life in it, however, the PC is still stupid at first. It has some primitive sense of self as it checks to see what parts are installed and working, like those patients who've awakened from a coma and check to be sure they have all their arms and legs and that all their joints still work. But beyond taking inventory of itself, the newly awakened PC still can't do anything really useful; certainly nothing we would even remotely think of as intelligent.

At best, the newly awakened PC can search for intelligence—intelligence in the form of an operating system that gives structure to the PC's primitive, amoebic existence. Then comes a true education in the form of application software—programs that tell the PC how to do tasks faster and more accurately than we could. The PC becomes a student who has surpassed its teacher.

But not all kinds of computers have to endure such a torturous rebirth each time they're turned on. You encounter daily many computers that spring to life fully formed at the instant they're switched on. You might not think of them as computers, but they are: calculators, your car's electronic ignition, the timer in the microwave, and the unfathomable programmer in your DVR. The difference between these and the big box on your desk is hard-wiring. Computers built to accomplish only one task—and they are efficient about doing that task—are hard-wired. But that means they are more like idiot savants than sages.

What makes your PC such a miraculous device is that each time you turn it on, it is a tabula rasa, capable of doing anything your creativity—or, more usually, the creativity of professional programmers—can imagine for it to do. It is a calculating machine, an artist's canvas, a magical typewriter, an unerring accountant, and a host of other tools. To transform it from one persona to another merely requires setting some of the microscopic switches buried in the hearts of the microchips, a task accomplished by typing a command or by clicking with your mouse on some tiny icon on the screen.

**1944**
Harvard University and IBM develop the Mark 1, which uses IBM punched cards.

**1945**
John von Neumann describes a general purpose electronic digital computer with a stored program.

**1948**
ENIAC scientists create Electronic Control, the first computer firm, and begin to build UNIVAC for Census Bureau.

**1949**
Popular Mechanics predicts: "Computers in the future may weigh no more than 1.5 tons."

**1951**
UNIVAC delivered to U.S. Census Bureau three years late. It uses magnetic tape for input instead of punched paper.

**1952**
A complaint is filed against IBM, alleging monopolistic practices in its computer business.

**1952**
UNIVAC predicts landslide victory for Eisenhower on CBS. Human forecasts predict tight race. UNIVAC wins.

**1954**
Texas Instruments announces the start of commercial production of silicon transistors.

**1954**
IBM brings out 650, the first mass-produced computer. It's a great success, with 120 installations in first year.

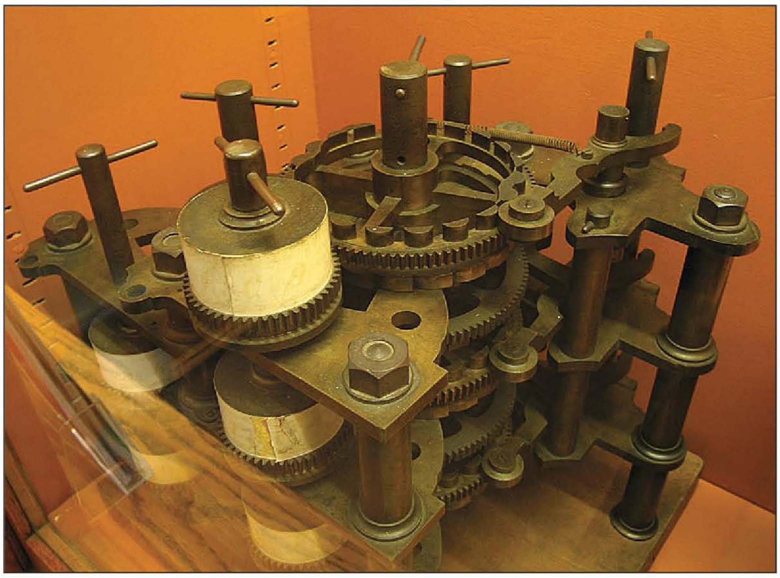
**1956**
Massachusetts Institute of Technology builds the first transistorized computer.

**1958**
Control Data Corporation introduces Seymour Cray's 1604. At $1.5 million, it's half the cost of the IBM computer.

**1958**
Jack Kilby completes first integrated circuit, containing five components on a single piece of silicon.

Such intelligence is fragile and short-lived. All those *millions* of microscopic switches are constantly flipping on and off in time to dashing surges of electricity. All it takes is an errant instruction or a stray misreading of a single switch to send this wonderful golem into a state of catatonia. Or press the Off switch and what was a pulsing artificial life dies without a whimper.

Then the next time you turn it on, birth begins all over again.



This portion of the Difference Engine #1, a forerunner to Charles Babbage's Analytical Engine--the first true computer--was completed in 1821. It contained 2,000 handmade brass parts. The entire machine would have used 25,000 parts and would have weighed 3 tons. The Analytical Engine was never completed, although part of it was built by Babbage's son, Henry, in 1910, and was found to be "buggy."

***Courtesy of IBM***

## How Computers Used to Work

At the beginning of the 21st century, computers are such complex beasts—despite their relative youth—that it's difficult to imagine how such elaborate contraptions could have sprung fully grown from the brows of their creators. The explanation is, of course, that they didn't. The development of computers has been an evolutionary process, and often it's well nigh impossible to figure out which came first, the chicken of software or the egg of hardware.

Human attempts to create tools to manipulate data date back at least as far as 2600 B.C. when the Chinese came up with the abacus. Later on, Leonardo da Vinci created a mechanical calculator. When the slide rule was invented in 1621, it remained the mathematician's tool of choice until the electronic calculator took over in the early 1970s.

All the early efforts to juggle numbers had two things in common: They were mechanical and they were on a human scale. They were machines made of parts big enough to assemble by hand. Blaise Pascal's Arithmetic Machine used a system of gears turned by hand to do subtraction and addition. It also used punch cards to store data, a method that's survived well into the 20th century.

**1960**
2,000 computers are in use in the United States.

**1965**
Digital Equipment Company's first successful minicomputer, the PDP-8. At $18,000, soon 50,000 are sold.

**1970**
Xerox creates the Palo Alto Research Center (PARC), which gave birth to many essential computer technologies.

**1971**
Intel's Ted Hoff designs 4004 chip, the first microprocessor. Price $200, with 2,300 transistors and 60,000 OPS.

**1973**
Architecture using CP/M operating system becomes the standard for the next eight years until MS-DOS is introduced.

**1975**
Popular Electronics announces the Altair 8800, the first personal computer.

**1975**
The first known use of the word Micro-soft appears in a letter from Bill Gates to his future partner, Paul Allen.

**1976**
Stephen Jobs and Steve Wozniak show first Apple computer at Home Brew Computer Club, later known as Silicon Valley.

**1977**
Radio Shack introduces the TRS-80 Model 1, lovingly referred to by its hobbyist fans as the Trash 80.

**1981**
IBM introduces its personal computer, which uses Intel's 16-bit 8086 processor.

**1982**
Compaq introduces the first IBM PC clone computer. Personal Computer is Time's "Man of the Year."

**1984**
Apple introduces the Macintosh, a computer using a mouse and graphic interface.

**1986**
Microsoft goes public at $21 a share, raises $61 million.

In 1830, Charles Babbage invented—on paper—the Analytical Engine, which was different from its predecessors because, based on the results of its own computations, it could make decisions such as sequential control, branching, and looping. But Babbage's machine was so complex that he died in 1871 without finishing it. It was built between 1989 and 1991 by dedicated members of the Science Museum in London. The physical size and complex mechanics of these mechanisms limited their usefulness; they were good for only a few tasks, and they were not something that could be mass produced.

In 1888, Herman Hollerith, the founder of what was to become IBM, created a machine that used punched cards to tabulate the 1890 U.S. Census. The device tabulated the results in six weeks instead of the seven years it had taken to compile the census by hand.

***Courtesy of Smithsonian Institute***

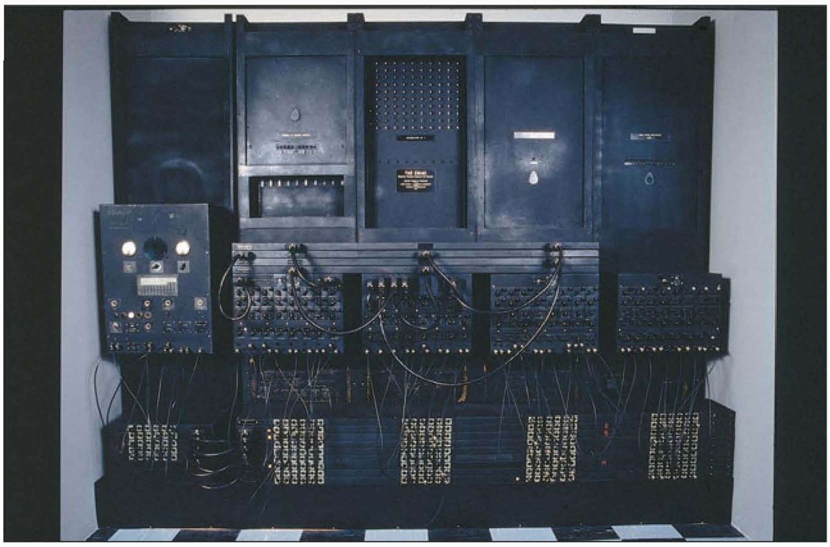
Mechanical devices of all types flourished modestly during the first half of the 20th century. Herman Hollerith invented a mechanized system of paper cards with holes in them to tabulate the U.S. Census. Later, in 1924, Hollerith's Computing-Tabulating-Recording Company changed its name to International Business Machines.

Although no one could have known it at the time, the first breakthrough to the modern computer occurred in 1904 when John Ambrose Fleming created the first commercial diode vacuum tube, something Thomas Edison had discovered and discarded as worthless. The significance of the vacuum tube is that it was the first step beyond the human scale of machines. Until it came along, computations were made first by gears and then by switches. The vacuum tube could act as a switch turning on and off thousands of times faster than mechanical contraptions.

Vacuum tubes were at the heart of Colossus, a computer created by the British during World War II to break the codes produced by the Germans' Enigma encrypting machine. And the Germans reportedly came up with a general-purpose computer—one not limited to a specific task as Colossus was. But the German invention was lost or destroyed in the war.

The war also gave birth to ENIAC (Electronic Numerical Integrator Analyzer and Computer), built between 1943 and 1945 by the U.S. Army to produce missile trajectory tables. ENIAC performed 5,000 additions a second, although a problem that took two seconds to solve required two days to set up. ENIAC cost $500,000, weighed 30 tons, and was 100 feet long and 8 feet high. It contained 1,500 relays and 17,468 vacuum tubes.

Those same tubes that made ENIAC possible in the first place were also its Achilles' heel. Consuming 200 kilowatts of electricity each hour, the tubes turned the computer into an oven, constantly cooking its own components. Breakdowns were frequent. What was needed was something that did the job of the tubes without the heat, bulk, and fragility. And that something had been around since 1926.



The ENIAC, built between 1943 and 1945, was the first all-electronic computer. It used so much power that legend says the lights of surrounding Philadelphia dimmed when the ENIAC was switched on.

***Courtesy of Smithsonian Institute***

In 1926, the first semiconductor transistor was invented, but it wasn't until 1947, when Bell Labs' William Shockley patented the modern solid-state, reliable transistor, that a new era in computing dawned. The transistor did essentially the same thing a vacuum tube did—control the flow of electricity—but it was the size of a pea and generated little heat. Even with the transistor, the few computers built then still used tubes. It wasn't until 1954, when Texas Instruments created a way to produce silicon transistors commercially, that the modern computer took off. That same year IBM introduced the 650, the first mass-produced computer. Businesses and the government bought 120 of them the first year.

Four years later, Texas Instruments built the first integrated circuit by combining five separate components and the circuitry connecting them on a piece of germanium half an inch long. The integrated circuit led to the modern processor and has made a never-ending contribution to smaller and smaller computers.

The computer grew increasingly smaller and more powerful, but its cost, complexity, and unswerving unfriendliness kept it the tool of the technological elite. It wasn't until 1975 that something resembling a personal computer appeared. The January issue of Popular Electronics featured on its cover something called the Altair 8800, made by Micro Instrumentation and Telemetry Systems (MITS). For $397, customers got a kit that included an Intel 8080 microprocessor and 256 bytes of memory. There was no keyboard; programs and data were both entered by clicking switches on the front of the Altair. There was no monitor. Results were read by interpreting a pattern of small red lights. But it was a real computer cheap enough for anyone to afford. MITS received orders for 4,000 Altair systems within a few weeks.

The first computer cheap enough for individuals to afford was the Altair 8800, created by a small New Mexico firm, MITS. It cost $397 without a keyboard or screen.

***Courtesy of The Computer Museum***

The new computer was at first a toy for hobbyists and hackers. They devised clever ways to expand the Altair and similar microcomputers with keyboards, video displays, magnetic tape, and then diskette storage. Then two hackers—Stephen Jobs and Steve Wozniak—created a personal computer that came complete with display, built-in keyboard, and disk storage, and began hawking it at computer clubs in California. They called it the Apple, and it was the first personal computer that was powerful enough,

and friendly enough, to be more than a toy. The Apple, along with computers from Radio Shack and Commodore, began appearing in businesses, sometimes brought in behind the backs of the people in white lab coats who ran the "real" mainframe computers in a sealed room down the hall. The information services—or IS, as the professional computer tenders came to be called–disparaged the new computers as toys, and at the same time they saw microcomputers as a threat to their turf.


The Apple, introduced in 1976, was an immediate hit partially because a program called VisiCalc, which did the math of an electronic ledger sheet, justified the computer as a business cost.

***Courtesy of Apple Corp.***

The development that finally broke the dam, unleashing microcomputers on a society that would forever after be different, was not a technical invention. It was a marketing decision IBM made when creating its first personal computer, the IBM PC. IBM wanted to keep the price down, and so it decided to build the computer from components that were already available off the shelf from several suppliers. IBM also made the overall design of the PC freely available to competitors. The only part of the machine IBM copyrighted was the BIOS, the crucial basic input/output system, computer code residing in a single chip that defined how software was to interact with the PC's hardware. The competition could create their own PCs as long as they duplicated the operations of IBM's BIOS without directly copying it.

While Apple continued to keep its design proprietary, IBM's openness encouraged the creation of IBM clones that could use the same software and hardware add-ons the PC used. And the clones, while competing with IBM, at the same time helped establish the IBM architecture as the machine for which software and add-on hardware developers would design.

## KEY CONCEPTS

**BIOS (basic input/output system)** A collection of software codes built into a PC that handle some of the fundamental tasks of sending data from one part of the computer to another.

**boot or boot-up** The process that takes place when a PC is turned on and performs the routines necessary to get all the components functioning properly and the operating system loaded. The term comes from the concept of lifting yourself by your bootstraps.

**circuit board** Originally, wires ran from and to any component in any electrical device, not just computers. A circuit board replaces the need for separate wiring with the metallic traces printed on the board—sometimes also on the bottom of the board and in a hidden middle layer. The traces lead to connections for processors, resistors, capacitors, and other electrical components. The importance of the circuit board is that its entire creation can be automated, and the board packs more components into an ever-smaller space.

**clock** A microchip that regulates the timing and speed of all the computer's functions. The chip includes a crystal that vibrates at a certain frequency when electricity is applied to it. The shortest length of time in which a computer can perform some operation is one *clock*, or one vibration of the clock chip. The speed of clocks—and therefore, computers—is expressed in megahertz (MHz). One megahertz is 1 million cycles, or vibrations, a second. Thus, a PC can be described as having a 200 or 300 MHz processor, which means that the processor has been designed to work with a clock chip running at that speed.

**CMOS** An acronym for *complementary metaloxide semiconductor*—a term that describes how a CMOS microchip is manufactured. Powered by a small battery, the CMOS chip retains crucial information about what hardware a PC comprises even when power is turned off.

**CPU** An acronym for *central processing unit*, it is used to mean the *microprocessor*—also, *processor*—which is a microchip that processes the information and the code (instructions) used by a computer. The "brains" of a computer.

**expansion slot** Most PCs have unused slots into which the owner can plug circuit boards and hardware to add to the computer's capabilities. Most slots today are personal computer interface (PCI) or it's next-generation sibling PCI-Express (PCI-E). One other slot, the accelerated graphics port (AGP), accepts a video card designed to move images out of memory quickly, although it is fast being replaced by PCI-E–you might also see shorter slots on older computers. These are industry standard architecture (ISA), the only type of slots on the first PC.

**motherboard** A sheet of plastic onto which metallic circuits have been printed and to the rest of the PC's components are connected. These components could be connected via a socket, such as with the CPU, a slot, as with graphics cards and memory modules or they may be built directly onto the motherboard, as with external ports, such as USB.

**operating system** Software that exists to control the operations of hardware. Essentially, the operating system directs any operation, such as writing data to memory or to disk, and regulates the use of hardware among several application programs that are running at the same time. This frees program developers from having to write their own code for these most basic operations.

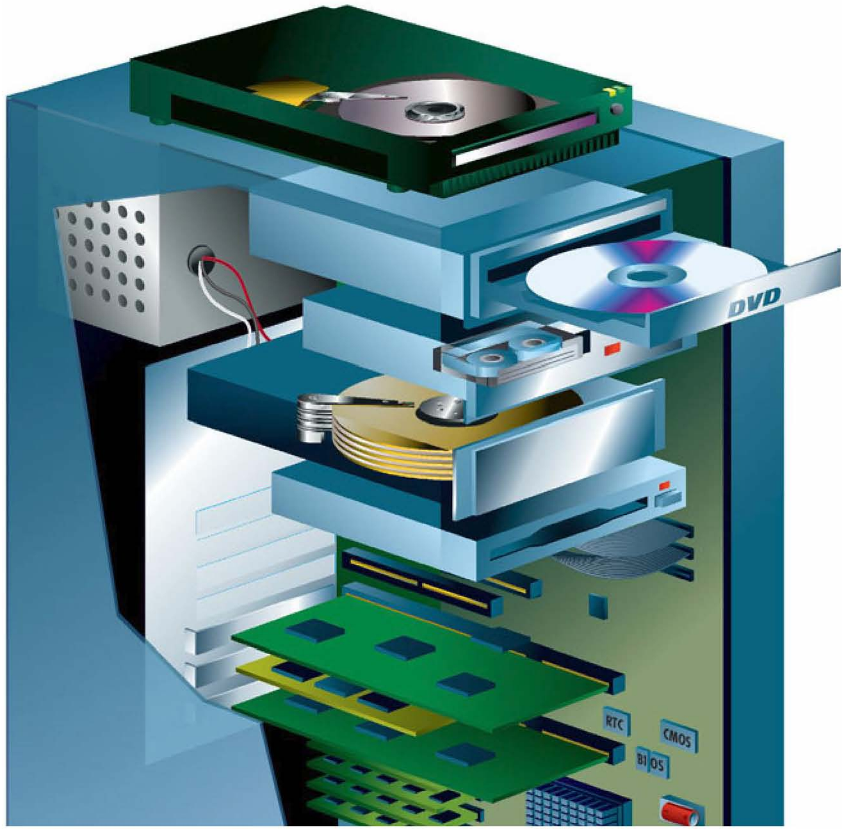
**ROM and RAM** Acronyms for Read Only Memory and Random Access Memory. ROM is memory chips or data stored on disks that can be read by the computer's processor. The PC cannot write new data to those chips or disk drives. RAM is memory or disks that can be both read and written to. Random access memory really is a misnomer because even ROM can be accessed randomly. The term was originally used to distinguish RAM from data and software that was stored on magnetic tape, and which could be accessed only sequentially. That is, to get to the last chunk of data or code on a tape, a computer must read through all the information contained on the tape until it finds the location where it stored the data or code for which it is looking. In contrast, a computer can jump directly to any information stored in random locations in RAM chips or on disk.

**system files** Small disk files that contain software code that are the first files a computer reads from disk when it is booted. The system files contain the information needed, following the initial hardware boot, to load the rest of an operating system.

**write and read** Writing is the process by which a computer stores data in either RAM chips or on a disk drive. Reading is the process by which a computer transfers data or software code from a drive to RAM or from RAM to the microprocessor.

CHAPTER 1

# Getting to Know the Hardware



THE handy thing about gears, pulleys, and the wheel—man's first tools—is that even someone who hasn't seen them before can quickly figure out what they are and how they're used. Humankind's latest and greatest tool—the computer—isn't nearly as obliging. It is made up of tiny rectangular blocks of plastic and cylinders of metal and ceramics that hide their inner workings. It has a maze of metallic lines, wires, and cables that would drive a lab rat schizo. The bigger components are encased in metal jackets that hide their purpose and operation. And the computer is as mute as a sphinx. The numbers and letters found on components certainly are not a part of any language you and I use.

But then, first meetings are always the most awkward. The purpose of this chapter is to get you and your PC beyond the sticky introductions so that you're on a first name basis with most of those mysterious components. As in any relationship, somebody's the boss. Let's be sure it's you by opening up your PC for a quick tour. All you'll need is a screwdriver. Check the screws sealing the case of your PC to determine what kind of screwdriver you'll need. Some PCs don't even require a screwdriver; instead they employ twistable thumbscrews that require no tools at all. A flashlight will also help. One of those little mirrors dentists use will enable you to see some of the more bashful components as well.

Now, before you do anything, touch a metal portion of the computer while it's still plugged in to discharge any static electricity that might have built up in your body and clothing. The components in your PC work with minute amounts of electrical current. A zap between your finger and some unsuspecting microchip after you've shuffled across a carpet on a cold, dry day is like sending the chip to the electric chair. You might not even feel the shock that kills your PC's components, so you should always take this precaution when working on your PC.

Then shut down your PC and unplug it. This is another safety precaution, but one to protect you instead of the machine. I don't know of anyone who has been fried by a PC, but don't take chances. Now you can undo the screws holding your case shut. They're usually around the edge of the back of your PC. The screws will run away if you're not careful, so put them somewhere where they can't get away from you. Slide off the chassis's cover. This is not always easy, but take some comfort from the fact that taking off the cover is easier than putting it back on.

Now shine your light into the bowels of the beast. What you're looking at is a mechanical organism of enormous complexity. If you counted the transistors in a modern PC as people, and the circuits connecting them as highways, you would be looking at something twice as enormous and intertwined as the United States—and you're the new kid in town. The illustration on the next two pages is a map to the major landmarks of a personal computer. It tells you what each does and why you would want it on your PC. Don't worry if you don't have everything shown in the picture. I've stuffed the example with more components than most computers have in the interest of thoroughness. The components in your PC might be located in different positions, but generally they will look like the components in the illustration. Following your tour of the inside of your PC, we'll linger at another illustration to get a close-up view of those highways of circuitry.

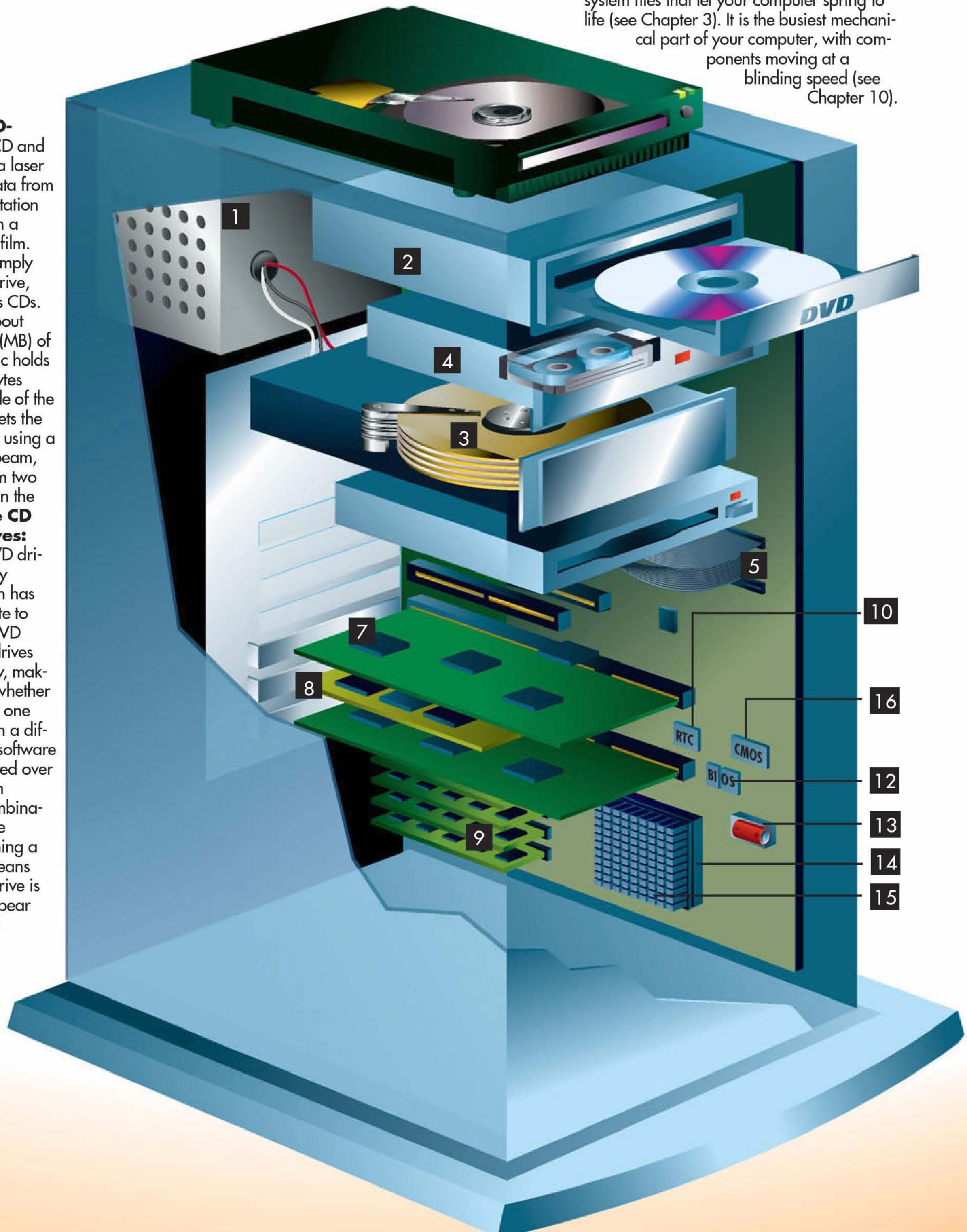
Finally, don't worry about damaging the computer. Unless you do something really stupid—such as throwing a glass of Coke into the computer while it's running—you won't hurt your PC. The only component that you could conceivably damage—if you don't count chips zapped by the homicidal static electricity—is the hard drive. The platters inside it are spinning at a breathtaking rate within fractions of a millimeter from other parts that could, if jostled, collide like race cars on Memorial Day. But don't worry. This is going to be a midnight run.

# Inside the Personal Computer

**1** **Power Supply:** All electricity enters your PC through this shielded metal box. Inside it, a transformer converts the current that comes from standard outlets into the voltages and current flows needed by various parts of the computer. All other components, from the motherboard to disk drives, get their power through the main supply via colored wires that end in plastic shielded connectors.

**2** **CD-ROM/DVD-ROM Drive:** CD and DVD drives use a laser beam to read data from a spiral of indentation and flat areas on a layer of metallic film. New PCs now simply feature a DVD drive, which also reads CDs. The CD holds about 650 megabytes (MB) of data. A DVD disc holds about 4.7gigabytes (GB) on each side of the disc. The DVD gets the extra storage by using a narrower laser beam, which reads from two separate layers in the DVD. **Writable CD and DVD drives:** Both CD and DVD drives are read-only devices, but each has versions that write to blank CD and DVD discs. Different drives record differently, making it uncertain whether a DVD made on one drive will play on a different drive. All software today is distributed over the Internet or on CD/DVD. In combination with writable CD/DVD becoming a standard, this means that the floppy drive is starting to disappear from the PC (see Chapter 12).

**3** **Hard Drive:** This is the main repository—in the form of magnetic recordings on hard, thin platters—of your programs and the documents on which you work. It also contains the system files that let your computer spring to life (see Chapter 3). It is the busiest mechanical part of your computer, with components moving at a blinding speed (see Chapter 10).



**4** **Floppy Drive:** Here you insert a 3.5-inch floppy disk (see Chapter 11). Most floppy disks hold 1.44 MB of data, the equivalent of 500 pages of typed, unformatted, double-spaced text—a short novel. It's also used to make backup copies of files in case something happens to the original files on the hard drive, but the size of hard drives and the universal inclusion of more capacious CD and DVD drives is driving floppies into extinction.(see Chapter 10).

**5** **Disk Controllers:** The motherboards of most new PCs have two types of connections for passing data and instructions to disk drives. The older IDE controller is used for floppy and optical drives, which are inherently slower that the controllers' ability to pass signals to the drives via flat, wide ribbons containing 40-80 wires. The newer Serial-ATA (SATA) connectors are reserved for hard drives, which take better advantage of the speed with which SATA passes information along a slim four-wire cable.

**6** **Expansion Slot:** Like disk controllers, expansion slots used to integrate new circuit boards into the motherboard, are combinations of the newest technology and legacy slots for compatibility with expansion boards still lagging behind in the engineering.

**7** **Video Card:** Translates image information into the varying electrical currents needed to display an image on the monitor (see Chapter 17).

**8** **Sound Card:** Contains the circuitry for recording and reproducing multimedia sound. This might be an expansion card or some computers might have it built into a few chips on the motherboard and attached by cables to external connections for amplified speakers, headphones, microphone, and CD player input (see Chapter 22).

**9** **RAM:** Random Access Memory is a collection of microchips aligned on small circuit boards that fit into slots with a couple of hundred or more connectors. RAM is where the computer stores programs and data while it uses them. When the computer is turned off, the contents of RAM are lost (see Chapter 4) .

**10** **Real-Time Clock:** A vibrating crystal in this component is the drummer that sets the pace and synchronizes the work of all the other components (see Chapter 2).

**11** **CMOS:** This is a special type of memory chip that uses a small battery to retain information about your PC's hardware configuration even while the computer is turned off (see Chapter 2).

**12** **BIOS:** If the microprocessor is your PC's brains, this is the heart. It is one or two chips that define the personality, or individuality, of your computer. The BIOS (*Basic Input/Output System*) knows the details of how your PC was put together and serves as an intermediary between the operating software running your computer and the various hardware components (see Chapter 3).

**13** **CMOS Battery:** Rarely needs changing, but if you ever have to, be sure you have a file backup of the information the CMOS chip contains (see Chapter 2).

**14** **Microprocessor:** Often called the brains of a computer, the microprocessor or central processing unit (CPU) is a tight, complex collection of transistors arranged so that they can be used to manipulate data. The processor handles most operations of your computer, the design of which dictates how software must be written to work correctly (see Chapter 5).

**15** **Heat Sink:** Because microprocessors produce so much heat, a heat sink is used to dissipate the heat so that internal components of the chip don't melt.

**16** **Fan:** A fan built into the power supply draws cool air over the heat-critical components inside the case. Be sure the opening to the fan is not blocked.

**17** **USB Ports:** *Universal serial bus* ports are a solution to PCs' lack of interrupts and other system resources to let software connect directly to peripherals. USBs can connect keyboards, input devices (mice, trackballs, etc.), flash memory drives, printers, and other devices without encountering resource conflicts (see Chapter 16) .


**18** **Mouse Port:** Also called a *PS2* port, this is a standard, but waning, feature on all current PCs. Personal computers can use a mouse that connects to a serial port or USB port.

**19** **Keyboard Port:** Keyboards are usually separate from the CPU housing and connect to a mini-DIN port, which looks identical to the PS2 port. The keyboard connection might be a larger, 5-pin round port on older systems and a USB port on newer systems.

**20** **Network Connector:** The network connecter allows you to connect your PC to a local area network (LAN) or a broadband cable or DSL modem for high-speed Internet access (see Chapter 27).

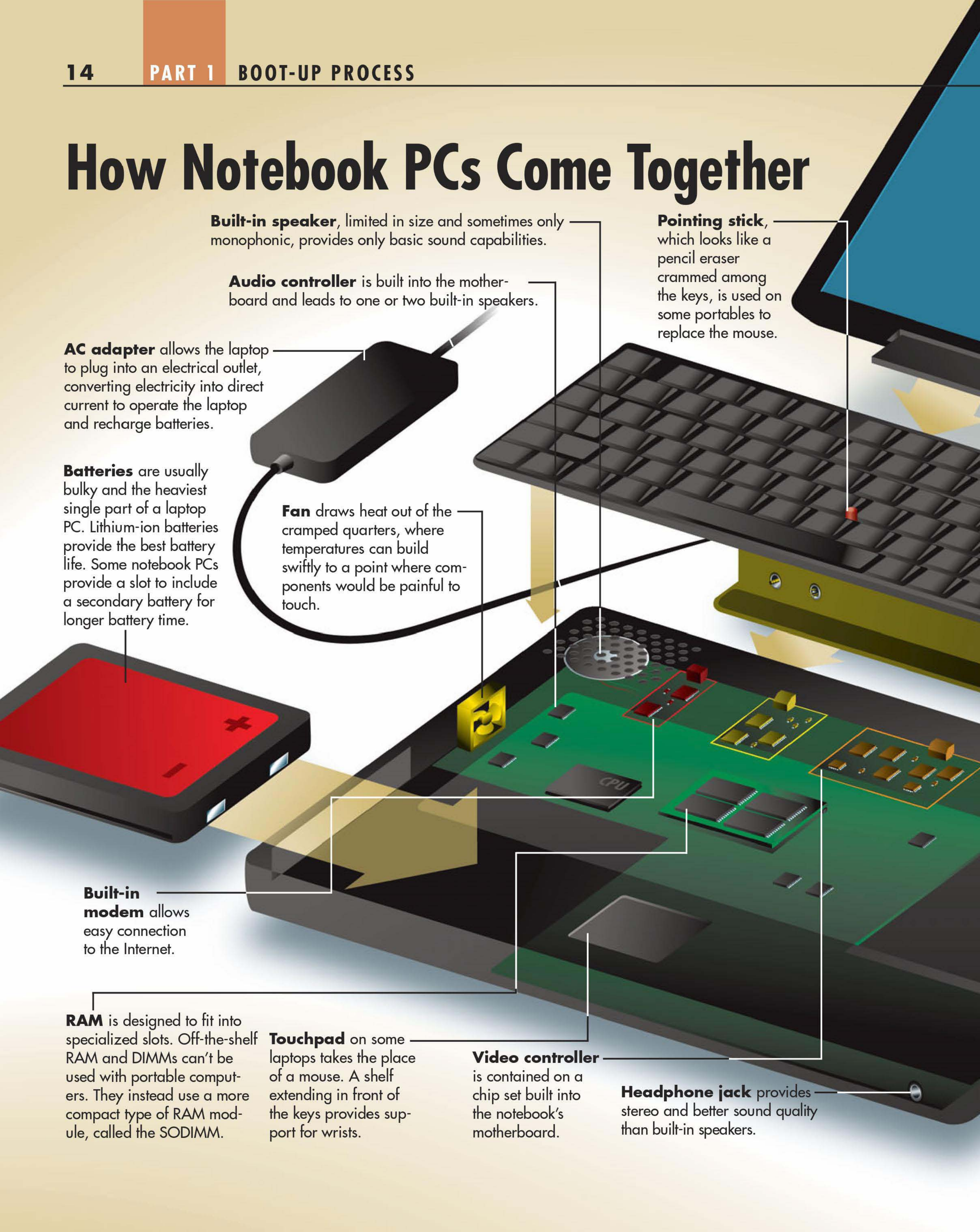
**21** **Parallel Port:** Although falling into disuse, when the parallel port is used, it's most often to connect a printer, but some drives and other peripherals can piggyback on the port.

**22** **Serial Ports:** Some PCs still have one or two serial ports, but they are all but obsolete because of the USB port. A PC can have four serial ports, but only two are usable at one time because one pair uses the same hardware resources as the other pair.

**23** **Sound Card Connections:** External jacks on the sound card or motherboard enable you to attach a microphone, speakers, or an external sound source. The PC's optical drive (CD or DVD) is attached to the sound card internally (see Chapter 22).

**24** **Modem:** Connects your PC to a telephone line so that you can get to information services and the Internet. Modems also come as external devices that connect to a serial port (see Chapter 26).

# How Notebook PCs Come Together



**Built-in speaker**, limited in size and sometimes only monophonic, provides only basic sound capabilities.

**Pointing stick**, which looks like a pencil eraser crammed among the keys, is used on some portables to replace the mouse.

**Audio controller** is built into the motherboard and leads to one or two built-in speakers.

**AC adapter** allows the laptop to plug into an electrical outlet, converting electricity into direct current to operate the laptop and recharge batteries.

**Batteries** are usually bulky and the heaviest single part of a laptop PC. Lithium-ion batteries provide the best battery life. Some notebook PCs provide a slot to include a secondary battery for longer battery time.

**Fan** draws heat out of the cramped quarters, where temperatures can build swiftly to a point where components would be painful to touch.

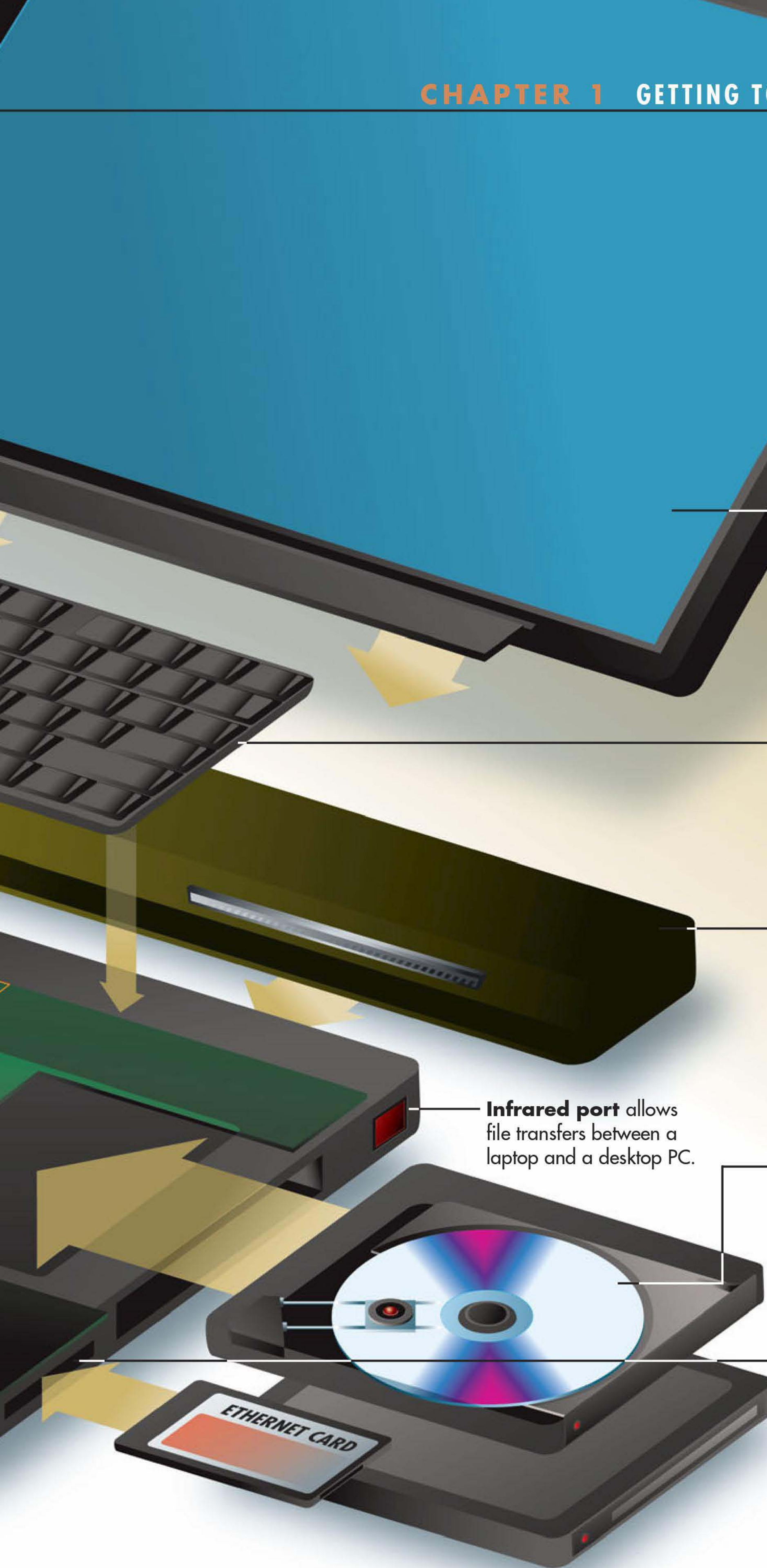
**Built-in modem** allows easy connection to the Internet.

**RAM** is designed to fit into specialized slots. Off-the-shelf RAM and DIMMs can't be used with portable computers. They instead use a more compact type of RAM module, called the SODIMM.

**Touchpad** on some laptops takes the place of a mouse. A shelf extending in front of the keys provides support for wrists.

**Video controller** is contained on a chip set built into the notebook's motherboard.

**Headphone jack** provides stereo and better sound quality than built-in speakers.




**Liquid crystal display (LCD)** , a screen type that eliminates the bulkiness of desktop cathode ray tube (CRT) monitors, is **backlit** by a panel that shines light through liquid crystal cells toward the user.

**Keyboards** might have smaller keys to fit them all on a laptop PC's surface. The keys might also have a shorter **travel**, the distance they can be depressed.

**External bus** is used to connect portable PC to a **docking station**, where it can access ordinary desktop monitor, keyboard, mouse, hard drive, and printer. In addition, separate jacks on the back allow direct connection to mouse, printer, external modem, monitor, and keyboard.

**Infrared port** allows file transfers between a laptop and a desktop PC.

**Floppy drive and Optical drive** often are combined into one component, or they can be separate components that take turns using a **drive bay**. Most portables include a DVD-ROM/CD-RW drive, although many high-end notebook systems include a rewritable DVD drive.

**PC card slot** allows other peripherals to be added easily. A PC card can serve the functions of a hard drive, modem, wireless network connection, or connection to an external CD-ROM drive. Many laptops have two PC card slots.

# How Tablet PCs Set You Free

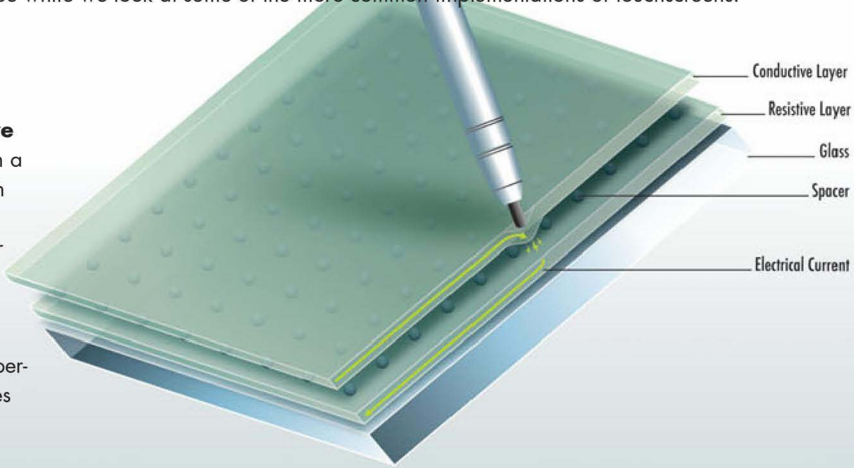
The deskless PC has long been computing's holy grail, a computer that slips the bounds of keyboard and cables, of monitor and mouse, to soar free as a bird—a bird endowed with a constant feed from the Internet, applications that will run a multinational corporation, and an easy abandon that lets it go with you anywhere, like a parrot perched on a pirate in pinstripe. Two factors foiled that quest—networking, which required an electronic umbilical cord, and haphazard human handwriting, which required a keyboard for intelligible communications. After a few false starts, the grail overflows with WiFi for always-on Internet and handwriting recognition that doesn't require you to learn a new way of writing.

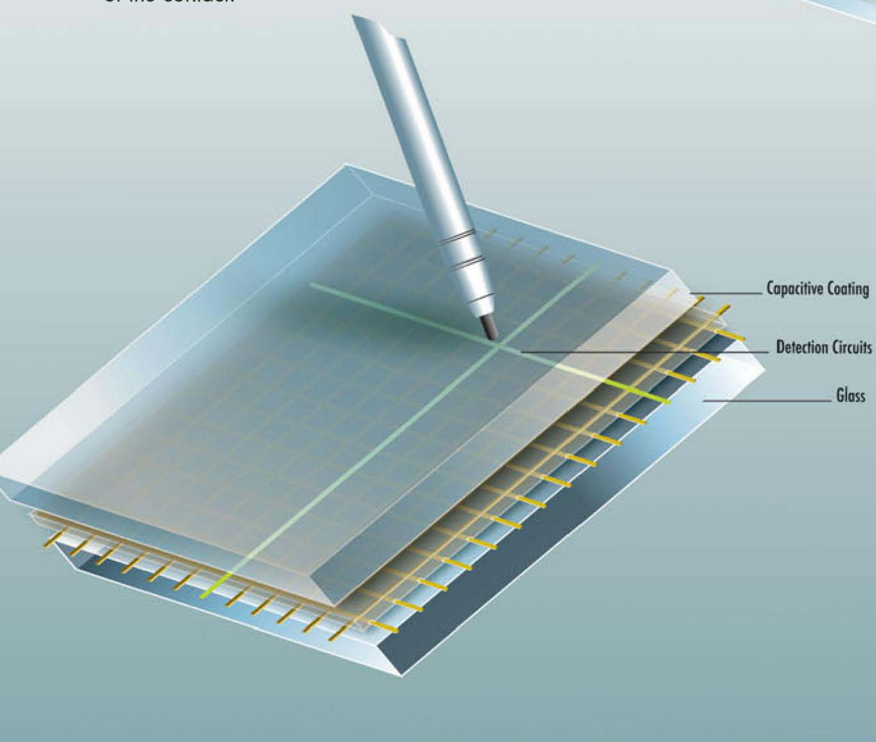
## How Screens Feel Your Touch

Of the dozen or so ways to design touchscreens, most of them follow a similar scheme: Within or above the layers of glass and plastic that make up the tablet computer's LCD screenething—electrical capacitance, electrical resistance, light, infrared light, or sound waves. When a stylus or finger disrupts whatever makes up the field, sensors measure the disruption and calculate where it must have happened. There are a couple of exceptions to this, as we'll see while we look at some of the more common implementations of touchscreens.

### Resistive System

The screen on a tablet PC using the **resistive system** includes a glass panel covered with a thin metallic layer made of a substance, such as indium tin oxide, that conducts electricity. Spacers on the layer support a metallic layer that resists the flow of electricity. When you touch the screen with your finger or a stylus, the two layers make contact, changing the electrical field produced by the layers. This permits the computer to calculate the coordinates of the contact.



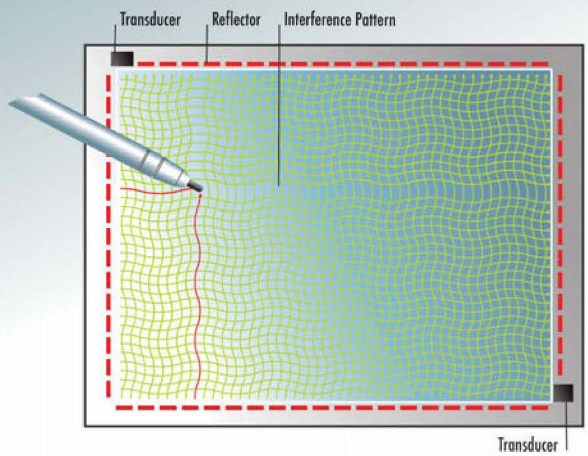



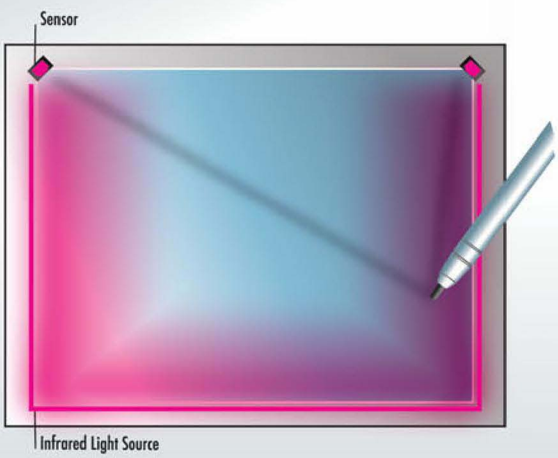

### Capacitive Touchscreen

In a **capacitive system**, a transparent material that stores an electrical charge covers the screen's glass panel. When you touch the monitor, some of the charge is transferred to your finger and the capacitive layer's charge decreases. Circuits at each corner of the screen constantly measure the change in the charge reaching them. From those measurements, the computer calculates where the touch occurred.

## Surface Acoustic Wave

A tablet PC using a **surface acoustic wave system** has two **transducers**, such as piezoelectric devices, that translate energy into movement and back again. Placed on the X and Y axes of the screen, one transducer converts electrical signals into ultrasonic vibrations that travel through the screen to the second transducer, which converts the vibrations back into electrical current. Also on the screens are reflectors, which reflect the vibrations. Because the reflected vibrations travel different paths, they arrive at the receiving transducer **out of phase**, producing an interference pattern. Something touching the screen's face also produces an interference pattern. The computer deciphers the changes in the signals' wave pattern to determine exactly where the screen was touched.



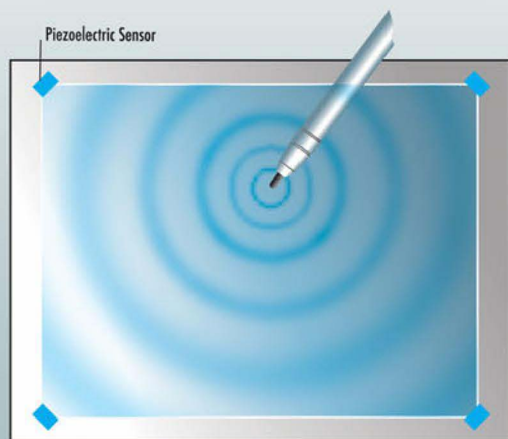




## Optical Imaging

Two or more simple **image sensors** overlook the screen, usually from the corners. Opposite the sensors are infrared backlights positioned so the sensors constantly detect the infrared lights. A touch on the screen is detected as a shadow by the sensors. By coordinating the position of the shadow as detected by each senor, a signal processor built into the screen determines where the touch took place.

## Dispersive Signal Sensing

Touchscreens equipped with **dispersive signal sensing** technology work without special layers or elements that emit a field of signals to be disrupted by a touch. Instead, this system measures the strengths of **bend waves** produced in the screen's glass surface by the pressure placed on the screen when it is touched. As the waves **disperse** through the glass, they become **smeared**—the faster high frequencies separate from the slower low frequencies. The farther the waves travel, the more they are smeared. Piezoelectric sensors positioned at each corner convert the smeared waves into electrical impulses a processor uses to determine the position of the touch.

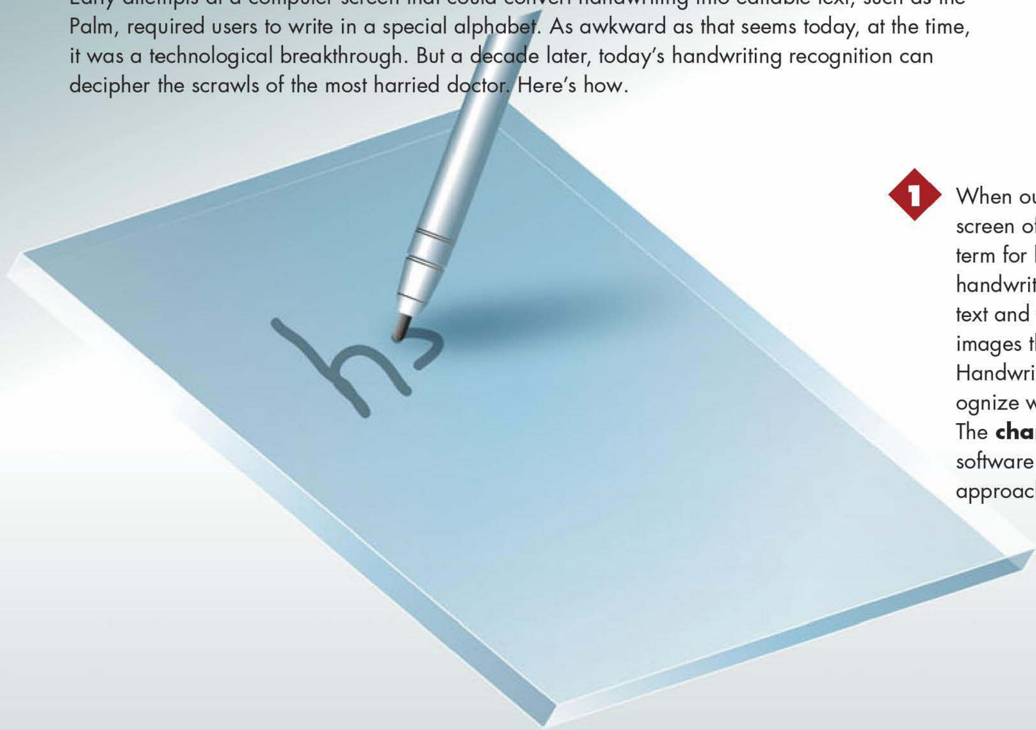



**What's Got the Touch?**

Not all objects register equally well on all types of touch-sensitive screens. Just about anything will work with a resistive system, provided it makes the two metallic layers touch each other. But a capacitive system registers only touches from objects that have their own electromagnetic fields. That can be your finger or a stylus that picks up an electrical charge from your hand. The wave system is similar to a resistor touchscreen except it does not accurately register hard, small objects such as the tip of a stylus. Both image sensing and dispersive signal processing have the advantage that no light-absorbing layers are embedded in the screen, making the screen's image brighter.

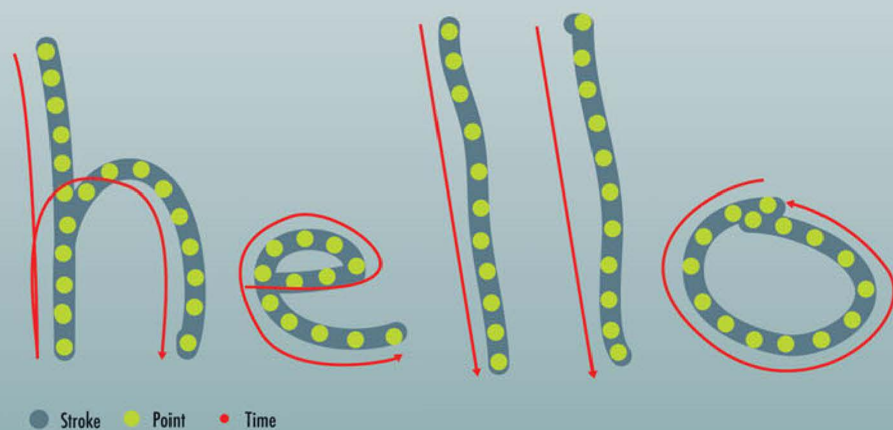
# How Tablet PCs Read Your Handwriting

Early attempts at a computer screen that could convert handwriting into editable text, such as the Palm, required users to write in a special alphabet. As awkward as that seems today, at the time, it was a technological breakthrough. But a decade later, today's handwriting recognition can decipher the scrawls of the most harried doctor. Here's how.



**1** When our hypothetical doctor writes on the screen of a tablet PC, he creates **ink**, the term for both a touchscreen's display of handwriting before it is turned into digital text and for drawings, arrows, and other images that should not be turned into text. Handwriting recognition's first job is to recognize what is handwriting and what isn't. The **character expert** in the recognition software begins with a two-pronged approach.

**2** One approach is **stroke timing**. An ink pattern is interpreted as a sequence of points on a flat surface. Their sequence is based on the timing with which each point appeared. The ink not only has a shape, but it has a trajectory that aids in determining what the ink represents. A lowercase "g" and "q" can both contain closed loops whose distinctions are too subtle for a computer to recognize. But given the trajectories of the strokes in the two characters, a tablet PC can easily discern the two.

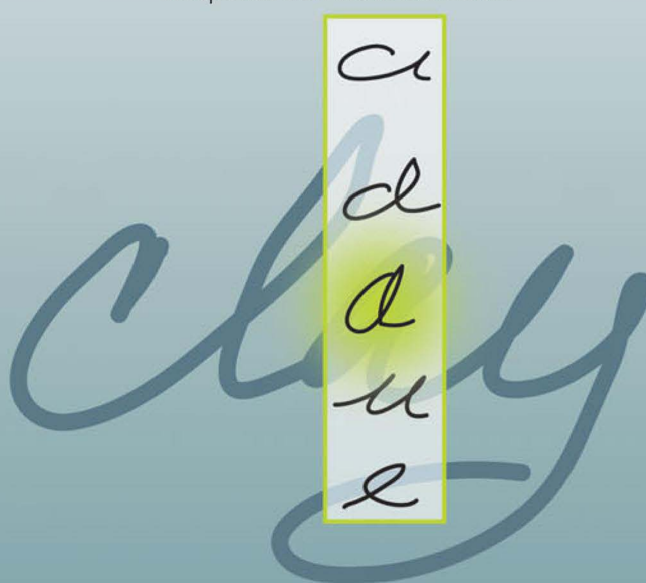



**3** The other approach is a **spatial matrix**. The recognition software plots the digital ink on a matrix with X and Y axes and a point of origin. The software compares the matrix plot to a database of shapes, any of which may be used to represent a specific letter, and selects the closest match.

**4** Because a tablet is designed to recognize script handwriting as well as printed block letters, a **segmentation expert** analyzes the ink to decide where a continuous stream of ink should be divided to yield individual characters. For example, if the doctor writes "clay" in script as one continuous application of ink, the work can be divided at least three ways.


**5** As in the example of "clay," there is no way to determine the correct translation of ink to text by visual examination alone. The recognition software relies on the **language expert** to identify the word by comparing it to an exhaustive database of words, lexicons, regular expressions, idioms, and language models. If only one choice emerges, the expert picks that choice unaided. If there is some ambiguity, the software presents the doctor with a list of alternative possibilities. This method of recognizing words by looking at them in context is more accurate than recognizing individual letters when the words written by the doctor are in the expert's database.

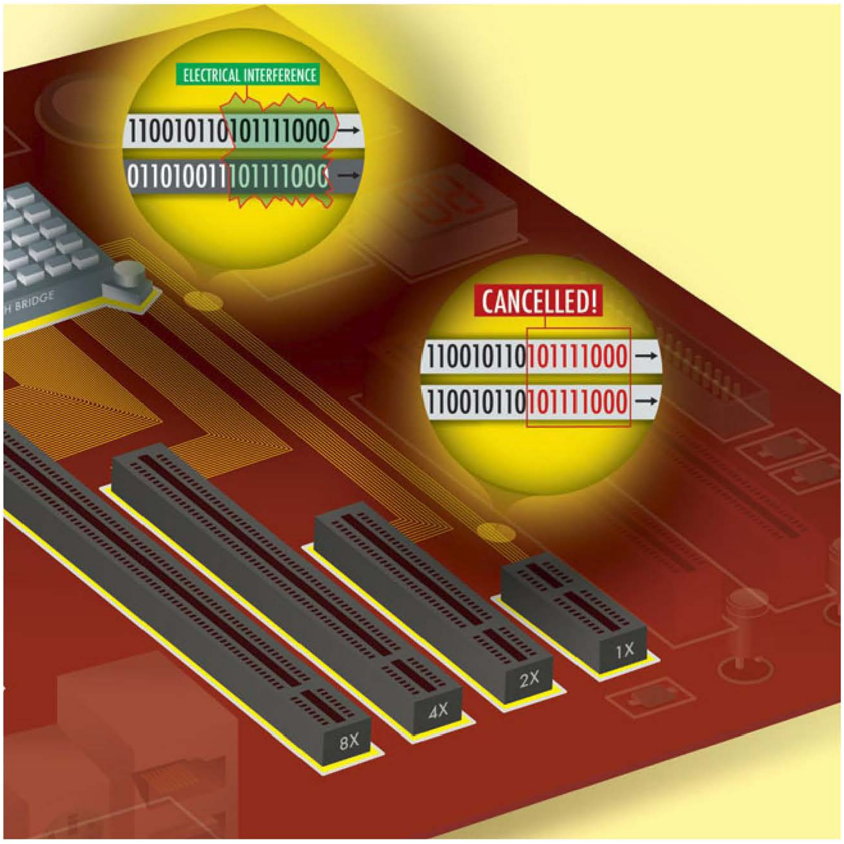

**6** In Windows XP, users could not train their computer to better recognize their handwriting. The only way to get better results was for users to retrain themselves to write more carefully. In Windows Vista, however, the handwriting engine gives users the option of correcting specific errors and then it learns from the corrections. The software also pays attention to the words used in email and incorporates them into its dictionaries.



CHAPTER

2

# How Circuits Juggle Data



**UNDER** the big top of your computer, the microprocessor—the central processing unit—is always the headliner. You don't see ads or reviews raving that a new PC has "revolutionary 100-ohm resistors!" Hard drives and graphics cards have the top supporting roles, but when it comes to the components on the motherboard—the mother of all boards—the CPU steals the spotlight.

There are many good reasons for the CPU's fame, but like all stars, it owes a lot to the little components—the circuit board supporting parts without which the central microprocessor would be only a cold slab of silicon. Without them, electronic messages meant for the CPU would crash into the chips and each other, moving so fast there would be no time to read their license plates. Contrarily, other messages would arrive like dying murder victims at the ER, so weak they can only whisper their crucial clues in pulses so faint the microprocessor can't understand them.

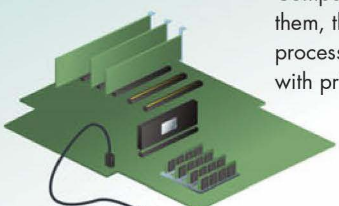
These supporting casts were much smaller in the early days of PCs because the starring role of the motherboard was much smaller. It was basically a platform for the microprocessor, which is a transportation grid for conveying signals back and forth between the CPU and the parts the CPU controlled—disc controller cards, video cards, sound cards, input/output cards. Back then, nearly everything that made a PC a PC was handled by expansion cards, which was handy because you could easily update a single component as innovation and budget allowed. Today, almost any computer comes with sound, video, disk controllers, and an assortment of input/output options all on the motherboard. Your computer's character is largely determined by the motherboard's capability, and those capabilities are largely defined by the parts that populate it.

So here, ladies and gentlemen, are the little parts that make it all possible.

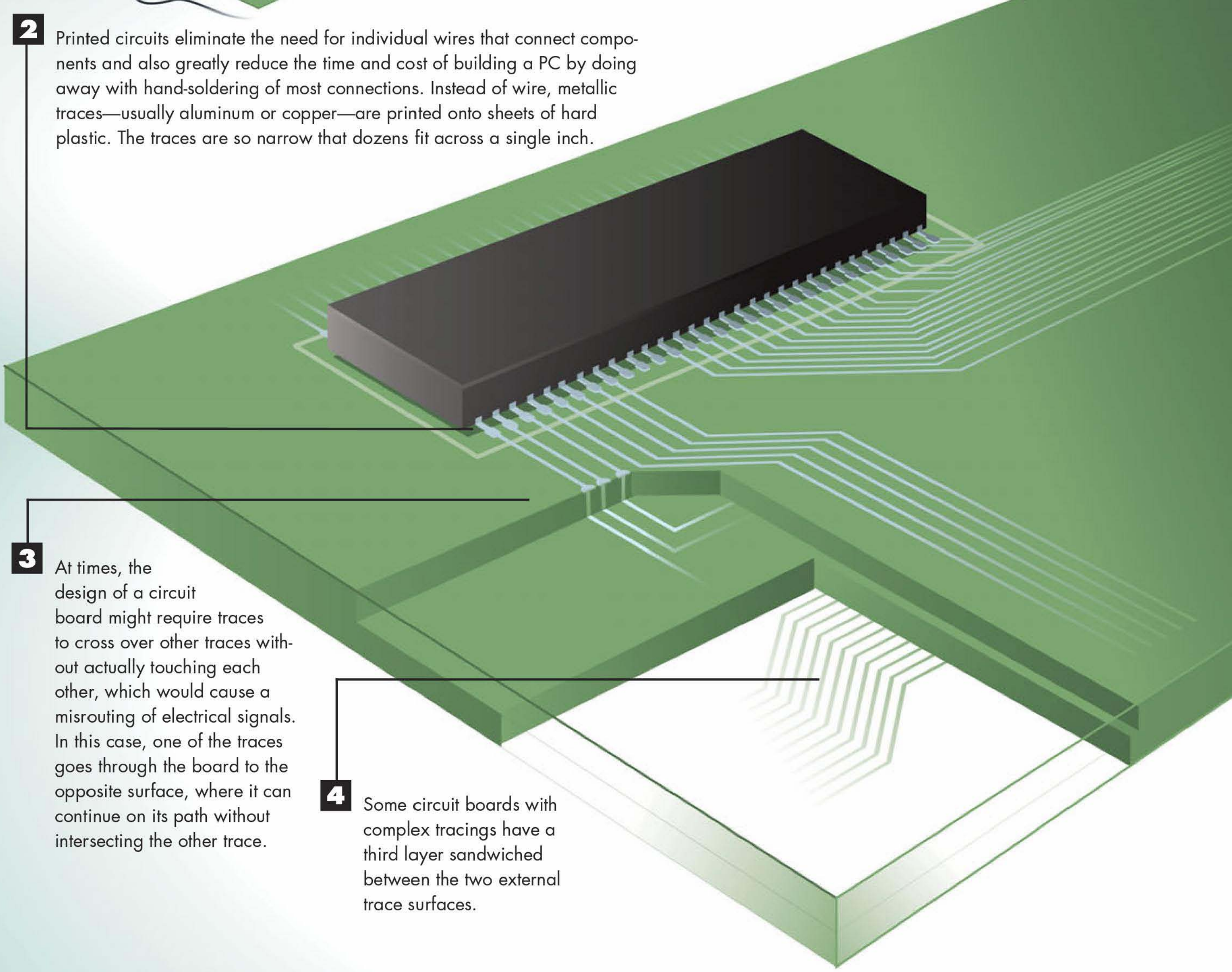
- Tiny metallic cans house the circuit board's strong men—the **resistors**! They clamp down on the wild, untamed electricity before it has the chance to burn up the rest of the components.
- Wrapped in ceramic casing and coats of plastic are the voracious, singing **capacitors**! They hum as they consume great quantities of electrical charge, holding it in so other components can have a steady supply or a sudden surge of electricity when they need it.
- Scattered everywhere on the motherboard are those mysterious, miniature monoliths, the **microchips**! What the millions of transistors do inside them is known to only a few.
- And connecting them all are stripes of copper and aluminum, **circuit traces,** that tie it all together so the individual players are a coordinated whole.

# How Circuit Boards Work

**1** Most of the components in a PC are mounted on printed circuit boards. The motherboard is the largest printed circuit. Expansion cards and memory chips plug into the motherboard. The memory chips are grouped together on small circuit boards to create **dual in-line memory modules**, or **DIMMs**. Components that at first glance don't appear to have circuit boards often have them, they're just hidden inside their housings. Disk drives and some microprocessors, such as the Athlon 64 and Core 2, tie their internal parts together with printed circuits.

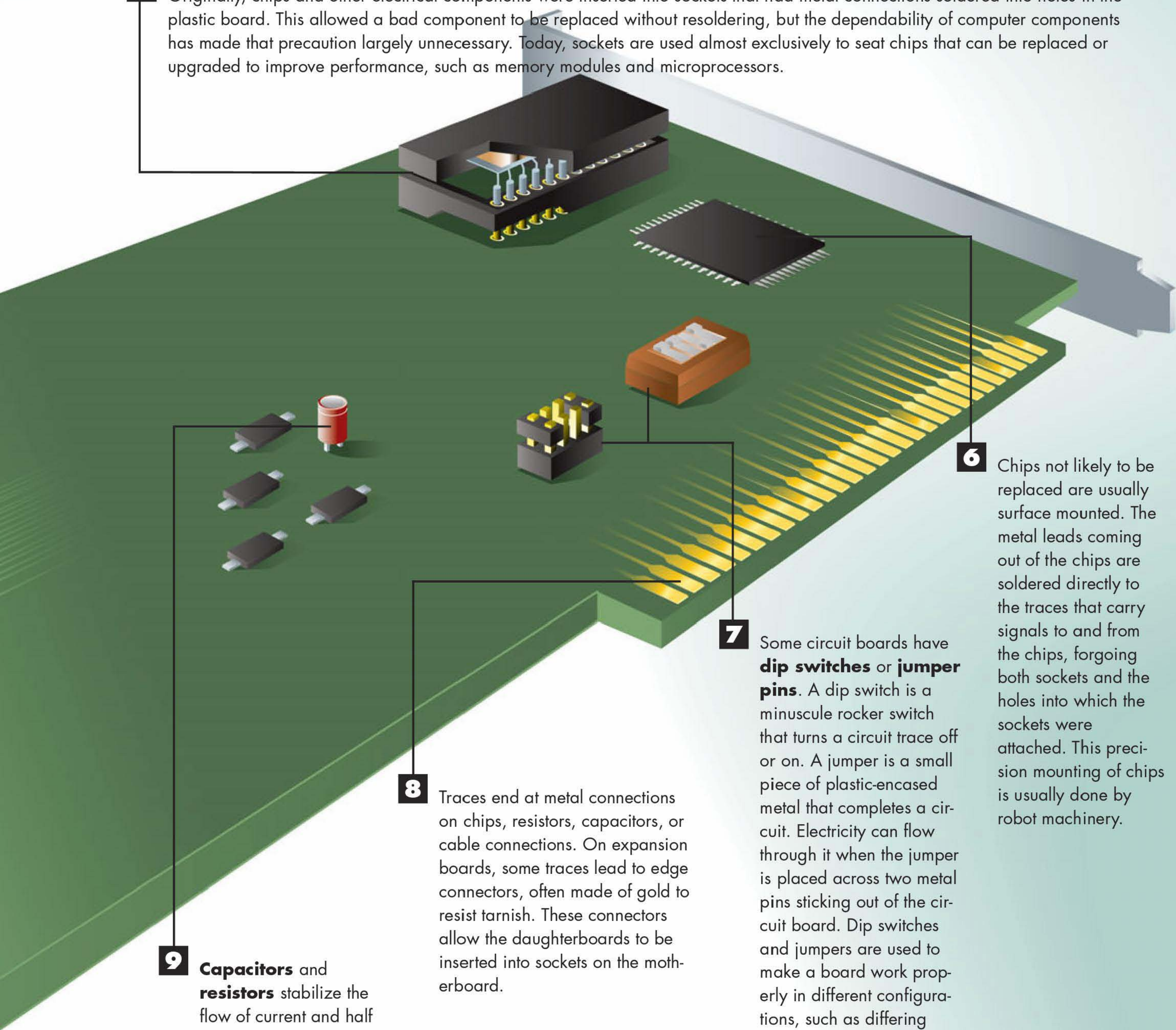


**2** Printed circuits eliminate the need for individual wires that connect components and also greatly reduce the time and cost of building a PC by doing away with hand-soldering of most connections. Instead of wire, metallic traces—usually aluminum or copper—are printed onto sheets of hard plastic. The traces are so narrow that dozens fit across a single inch.



**3** At times, the design of a circuit board might require traces to cross over other traces without actually touching each other, which would cause a misrouting of electrical signals. In this case, one of the traces goes through the board to the opposite surface, where it can continue on its path without intersecting the other trace.

**4** Some circuit boards with complex tracings have a third layer sandwiched between the two external trace surfaces.



**5** Originally, chips and other electrical components were inserted into sockets that had metal connections soldered into holes in the plastic board. This allowed a bad component to be replaced without resoldering, but the dependability of computer components has made that precaution largely unnecessary. Today, sockets are used almost exclusively to seat chips that can be replaced or upgraded to improve performance, such as memory modules and microprocessors.

**6** Chips not likely to be replaced are usually surface mounted. The metal leads coming out of the chips are soldered directly to the traces that carry signals to and from the chips, forgoing both sockets and the holes into which the sockets were attached. This precision mounting of chips is usually done by robot machinery.

**7** Some circuit boards have **dip switches** or **jumper pins**. A dip switch is a minuscule rocker switch that turns a circuit trace off or on. A jumper is a small piece of plastic-encased metal that completes a circuit. Electricity can flow through it when the jumper is placed across two metal pins sticking out of the circuit board. Dip switches and jumpers are used to make a board work properly in different configurations, such as differing amounts of memory.

**8** Traces end at metal connections on chips, resistors, capacitors, or cable connections. On expansion boards, some traces lead to edge connectors, often made of gold to resist tarnish. These connectors allow the daughterboards to be inserted into sockets on the motherboard.

**9** **Capacitors** and **resistors** stabilize the flow of current and half remove static and electrical surges or drops.

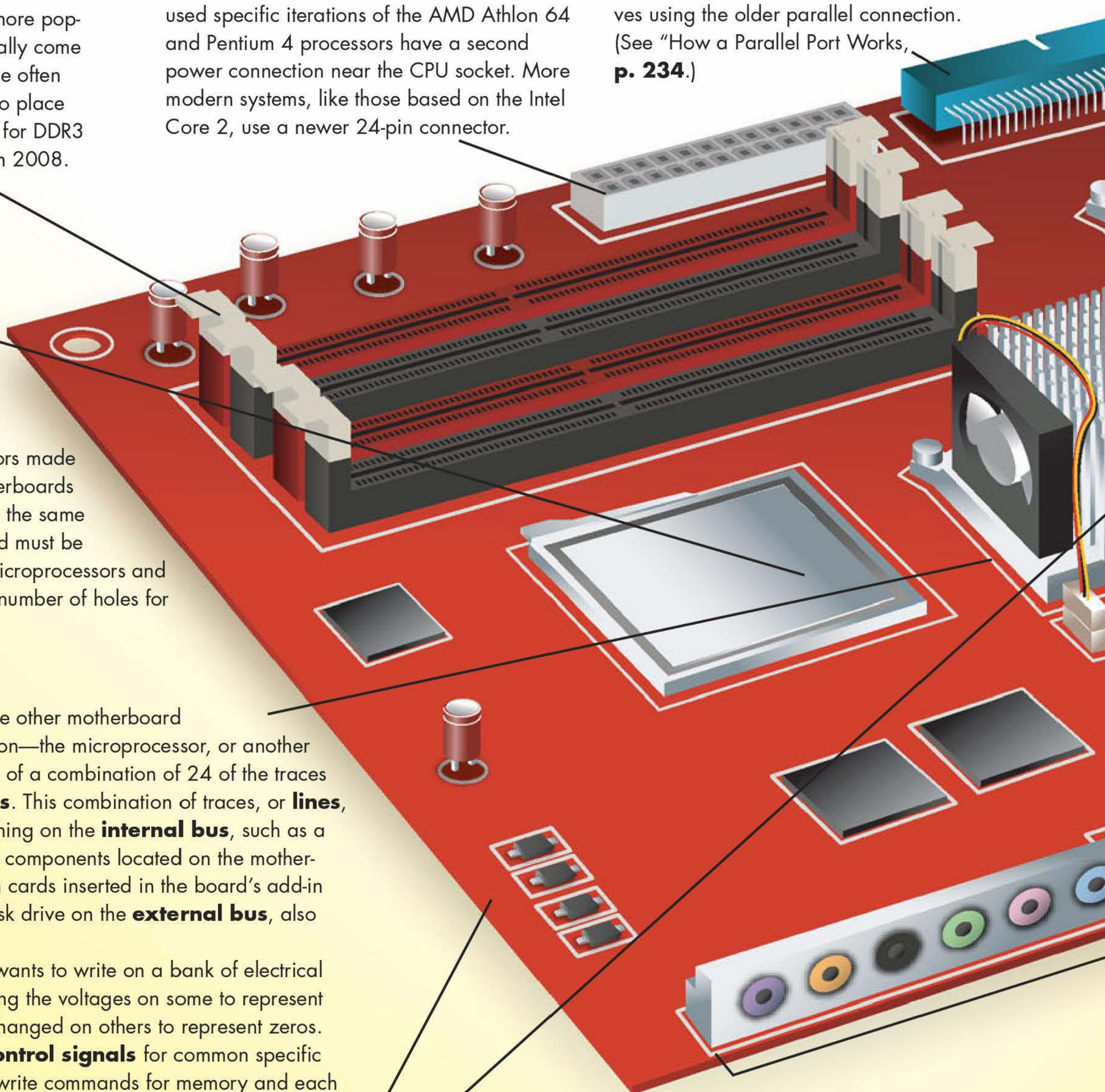
**10** **Pin connectors** are used by **ribbon cables**—wide, flat collections of wires joined together—for internal connections between circuit boards and disk drives.

# How the Motherboard Brings It All Together

**Memory Slots:** Current slots support either DDR (184 pins) or DDR2 RAM (240 pins), which is now the more popular type of memory. Slots usually come two or four to a board, and are often color-coded to tell you where to place matching memory cards. Look for DDR3 to start appearing in systems in 2008.

**Power Supply Connections:** Older boards have only one 20-pin connector. Boards that used specific iterations of the AMD Athlon 64 and Pentium 4 processors have a second power connection near the CPU socket. More modern systems, like those based on the Intel Core 2, use a newer 24-pin connector.

**IDE Connector:** Connects to two EIDE/ATA hard drives and optical drives using the older parallel connection. (See "How a Parallel Port Works," **p. 234**.)



**CPU Socket:** This determines what kind of **microprocessor**, or **central processing unit (CPU)**, the motherboard uses. Boards are designed to work with processors made by either **Intel** or **AMD**. Motherboards do not work with all CPUs from the same company. The socket and board must be designed for specific lines of microprocessors and must have the right shape and number of holes for the chip's pins to fit.

**Bus:** To send data to any of the other motherboard components—a **write** operation—the microprocessor, or another component, raises the voltages of a combination of 24 of the traces that make up the **address bus**. This combination of traces, or **lines**, is the unique address of something on the **internal bus**, such as a location in memory; one of the components located on the motherboard itself, such as expansion cards inserted in the board's add-in slots; or a device, such as a disk drive on the **external bus**, also called the **expansion bus**.
The processor puts the data it wants to write on a bank of electrical traces, the **data bus**, by raising the voltages on some to represent ones and leaving voltages unchanged on others to represent zeros. Other lines are used to pass **control signals** for common specific commands, such as read and write commands for memory and each input/output device.

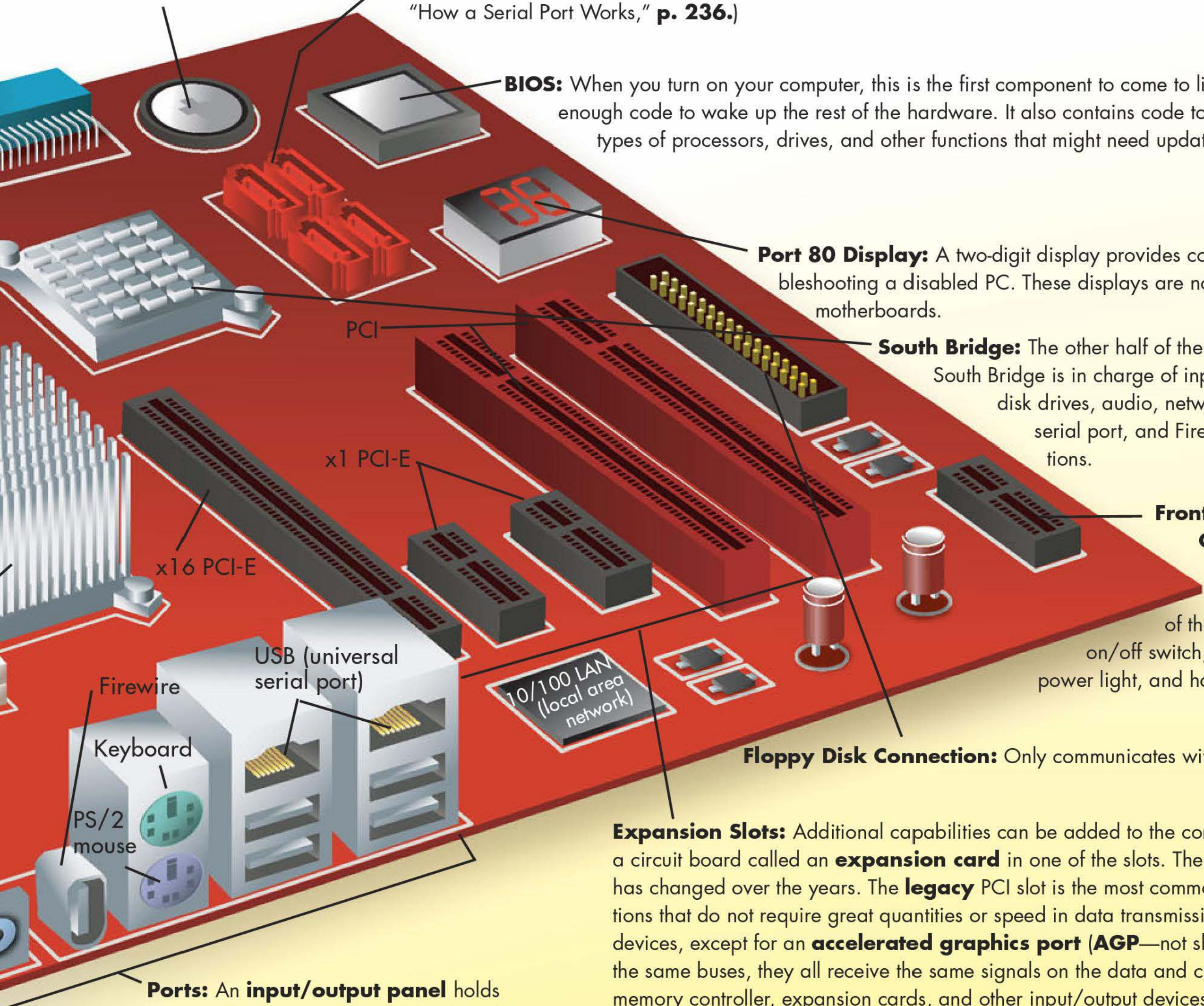
**The Motherboard:** As its name implies, the motherboard is the uniting element among all the chips and circuitry that make up a computer. Devices communicate with each other through the motherboard's circuits, from which they also draw their power. Motherboards come in different **form factors** that align the board with different sizes and styles of computer cases. They also come with different sockets that determine what types of chips and **expansion boards** they can accept.

**North Bridge:** The North Bridge and South Bridge together form the computer's **chip set**, secondary only to the processor in determining the performance and capabilities of a PC. The North Bridge chip either provides or controls the computer's graphics, RAM, and the **front side bus**, the main highway for data connecting graphics and memory to the CPU.

**Battery:** It keeps the BIOS chip alive.

**SATA Connectors:** Each connector, or **header**, is designed for the newer serial-ATA hard drives, providing faster delivery of drive data. These will eventually entirely replace IDE connections on most boards. (See "How a Serial Port Works," **p. 236.**)

**BIOS:** When you turn on your computer, this is the first component to come to life, providing enough code to wake up the rest of the hardware. It also contains code to support specific types of processors, drives, and other functions that might need updating occasionally.




**Port 80 Display:** A two-digit display provides codes used in troubleshooting a disabled PC. These displays are not found on all motherboards.

**South Bridge:** The other half of the PC **chip set**, the South Bridge is in charge of input/output with the disk drives, audio, networking, universal serial port, and Firewire communications.

**Front Panel Connectors:** Wires from these lead to the front of the PC for the on/off switch, reset switch, power light, and hard drive light.

**Floppy Disk Connection:** Only communicates with a floppy drive.

**Expansion Slots:** Additional capabilities can be added to the computer by plugging a circuit board called an **expansion card** in one of the slots. The design of the slots has changed over the years. The **legacy** PCI slot is the most common, used for functions that do not require great quantities or speed in data transmissions. Because all devices, except for an **accelerated graphics port** (**AGP**—not shown here) are on the same buses, they all receive the same signals on the data and control buses. The memory controller, expansion cards, and other input/output devices along the bus constantly monitor the command lines. When a signal appears on the write command line, for example, all the input/output devices recognize the command. The devices, alerted by the write command, turn their attention to the address lines. If the address specified on those lines is not the address used by a device, it ignores the signals sent on the data lines.

If the signals on the address lines match the address used by the adapter, the adapter accepts the data sent on the address lines and uses that data to complete the write command.

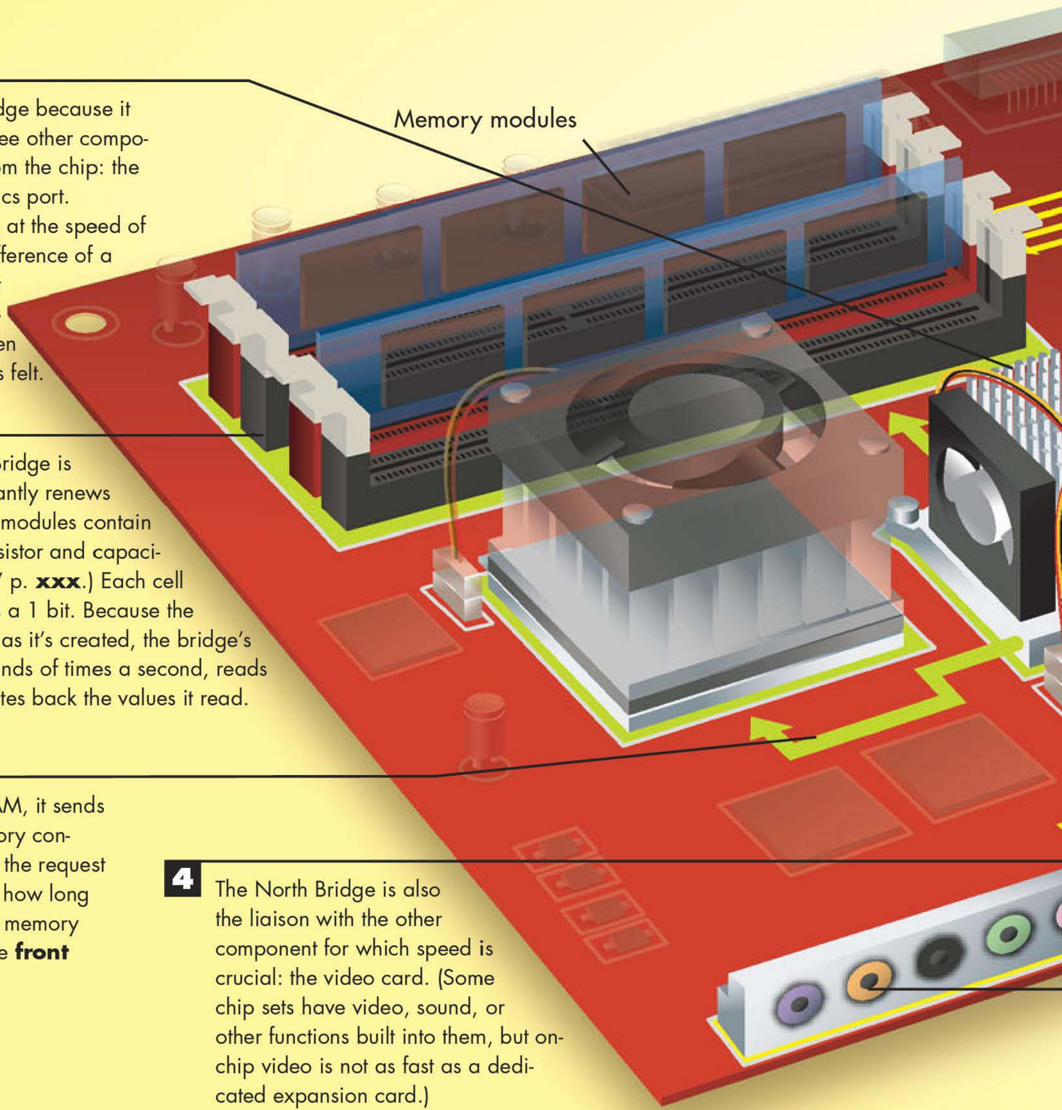
The **accelerated graphics port** is being pushed aside by the newer **PCI-Express** slots, which come in several denominations to make them the do-all, fit-all slot for every expansion board, not just graphics. The shorter ones here are **x1 PCI-E** shots and are common to all PCI Express slots. To handle graphics and sound data faster, the PCI-E slot can be expanded to **x4**, **x8**, or, shown here, **x16** slots, where the numbers represent multiples of the speed of an x1 PCI-E slot. Their ability to move data is indicated by the multiplier factor in their designations.

**Ports:** An **input/output panel** holds the miscellaneous ports on the back and front of the PC that are used for communicating with external devices. (Notice the lack of serial or parallel ports, which used to be standard. If they are needed for your computer's peripherals, they can be added with an expansion card.) Audio input/output ports are often part of the panel, although this board has a separate panel for them.

# How the North and South Bridge Move Traffic

The personal computer has become so complex that even the most recent, powerful processors can't do the entire job of managing the flow of data by themselves. The CPU has been given help in the form of the **chip set**, located nearby on the motherboard. The chip set traditionally consists of two microchips, often referred to as the **North Bridge** and the **South Bridge**, that act as the administrators to the CPU, or chief executive. The chip set bridges logical and physical gaps between the CPU and other chips, all the time watching and controlling the input and output of specific components. The exact function of the chip set is constantly changing. The bridges have been put into one chip and in some designs, the CPU reclaims some functions. But in all cases, the bridges determine what kinds of memory, processors, and other components can work with that particular motherboard. There is an unfortunate trend to replace the names North Bridge and South Bridge with less elegant terms such as **Graphics Memory Controller Hub (GMCH)** and the **I/O Controller Hub (ICH)**, even though their basic purpose is the same. For our purposes here, we'll stick to the more seemly bridge nomenclature.

## The North Bridge




**1** You can distinguish the North Bridge because it resides as close as possible to three other components that get special attention from the chip: the CPU, the memory, and the graphics port. Although for something operating at the speed of light, you wouldn't think that a difference of a couple of inches could matter. But when you're counting in nanoseconds—billionths of a second—even small differences make themselves felt.

**2** A crucial mechanism in the North Bridge is the memory controller, which constantly renews the memory modules (RAM). These modules contain memory cells, each made of a transistor and capacitor. (See "How a Transistor Works," p. **xxx**.) Each cell with an electrical charge represents a 1 bit. Because the charge begins to dissipate as soon as it's created, the bridge's memory controller endlessly, thousands of times a second, reads each of the millions of cells and writes back the values it read.

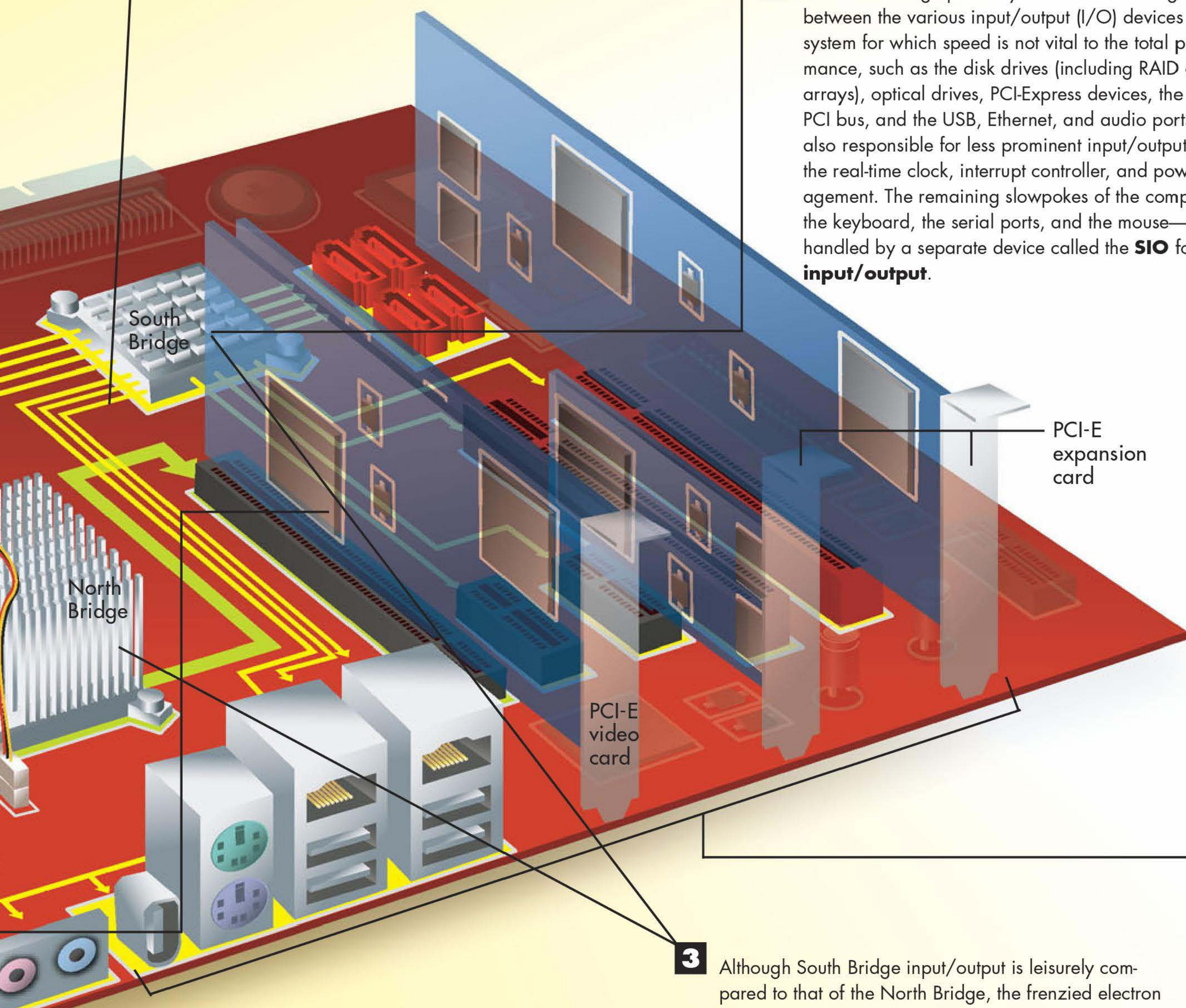
**3** When the CPU needs data from RAM, it sends a request to the North Bridge memory controller. The controller, in turn, sends the request along to memory and tells the CPU how long the processor must wait to read the memory over a speedy connection called the **front side bus (FSB)**.

**4** The North Bridge is also the liaison with the other component for which speed is crucial: the video card. (Some chip sets have video, sound, or other functions built into them, but on-chip video is not as fast as a dedicated expansion card.)

# The South Bridge

**1** The remaining connection of the North Bridge is to the South Bridge, ICH, or Input/Output Bridge, as the case may be.

**2** The South Bridge primarily handles the routing of traffic between the various input/output (I/O) devices on the system for which speed is not vital to the total performance, such as the disk drives (including RAID drive arrays), optical drives, PCI-Express devices, the older PCI bus, and the USB, Ethernet, and audio ports. It is also responsible for less prominent input/output, such as the real-time clock, interrupt controller, and power management. The remaining slowpokes of the computer—the keyboard, the serial ports, and the mouse—are handled by a separate device called the **SIO** for **super input/output**.



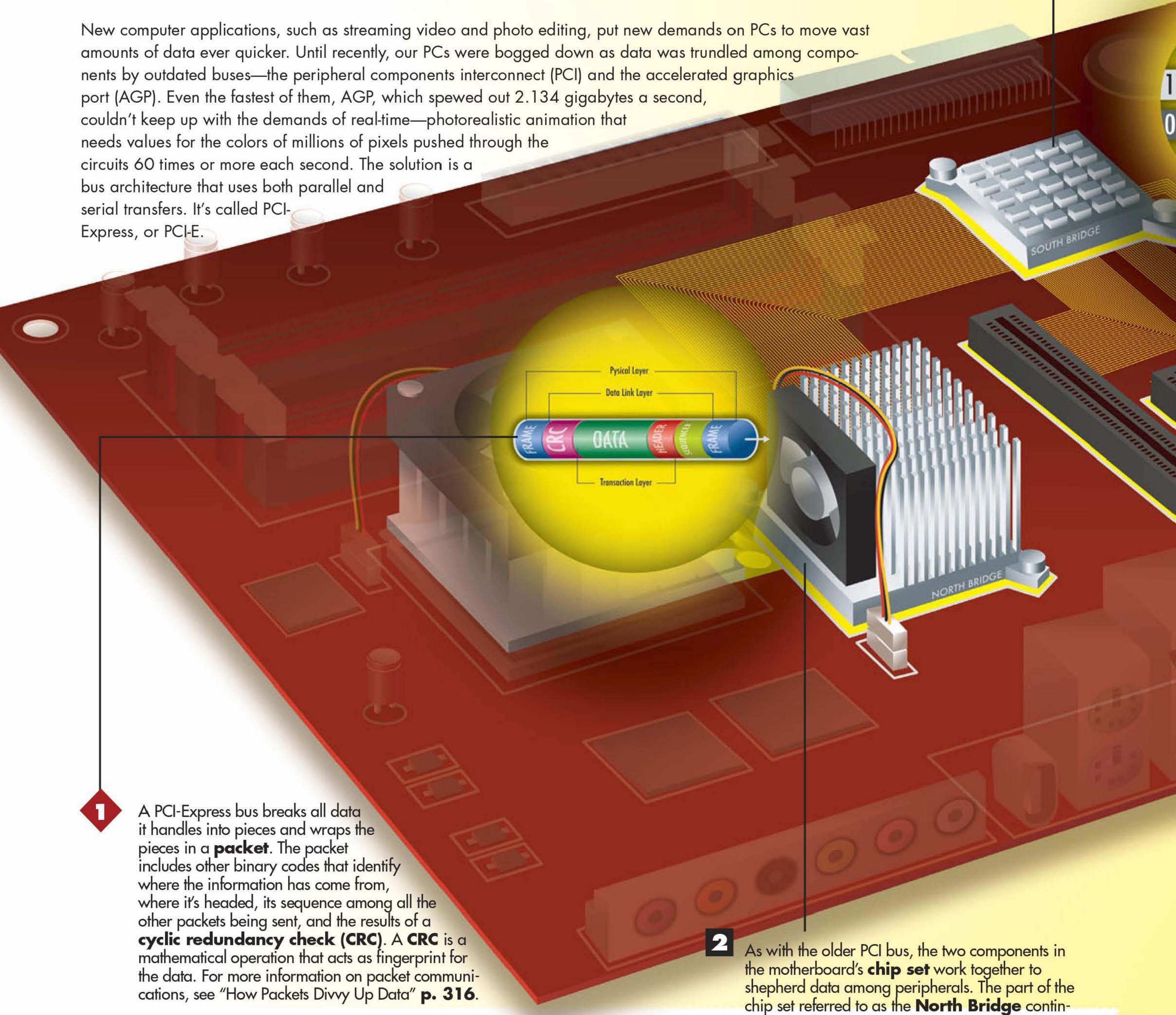

**3** Although South Bridge input/output is leisurely compared to that of the North Bridge, the frenzied electron traffic in both generates enough heat to require some sort of cooling device, such as a fan or heat sink, to stop the chips from overheating.

**4** Some South Bridge chips incorporate audio capabilities good enough to support Dolby Digital and THX multimedia audio.

**5** Previously the North Bridge worked with the accelerated graphics port (AGP), providing a quick transfer of bitmaps from RAM to the AGP card's own memory. Now, however, the still faster PCI-Express (PCI-E) interface is replacing AGP video as you'll see when you turn the page.

# How PCI-Express Breaks the Bus Barrier

New computer applications, such as streaming video and photo editing, put new demands on PCs to move vast amounts of data ever quicker. Until recently, our PCs were bogged down as data was trundled among components by outdated buses—the peripheral components interconnect (PCI) and the accelerated graphics port (AGP). Even the fastest of them, AGP, which spewed out 2.134 gigabytes a second, couldn't keep up with the demands of real-time—photorealistic animation that needs values for the colors of millions of pixels pushed through the circuits 60 times or more each second. The solution is a bus architecture that uses both parallel and serial transfers. It's called PCI-Express, or PCI-E.

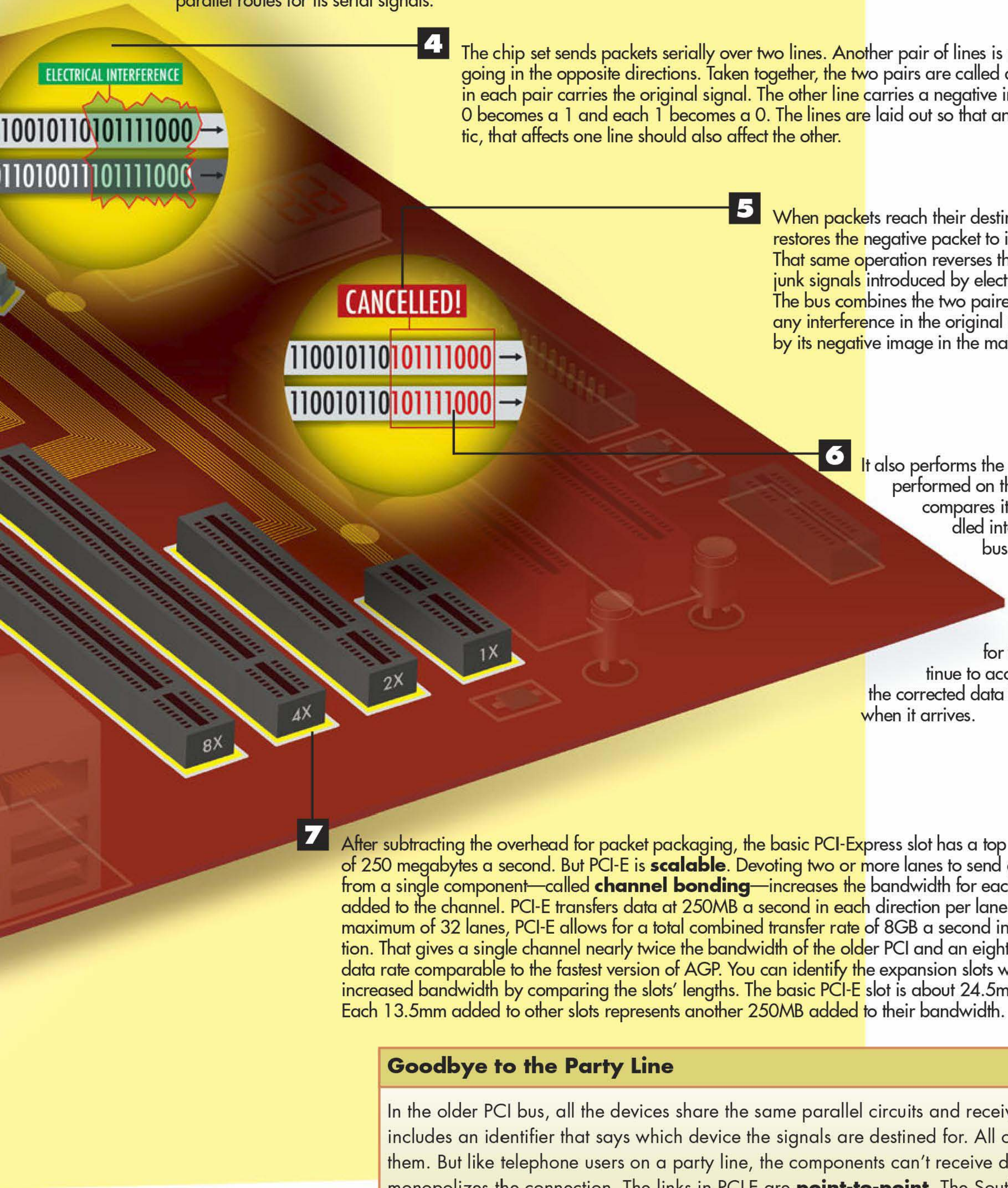



**1** A PCI-Express bus breaks all data it handles into pieces and wraps the pieces in a **packet**. The packet includes other binary codes that identify where the information has come from, where it's headed, its sequence among all the other packets being sent, and the results of a **cyclic redundancy check (CRC)**. A **CRC** is a mathematical operation that acts as fingerprint for the data. For more information on packet communications, see "How Packets Divvy Up Data" **p. 316**.

**2** As with the older PCI bus, the two components in the motherboard's **chip set** work together to shepherd data among peripherals. The part of the chip set referred to as the **North Bridge** continues to be the chip that rushes packets to the CPU and RAM, the components for which speed is most critical. It has traditionally passed less urgent packets to the **South Bridge** for handling.

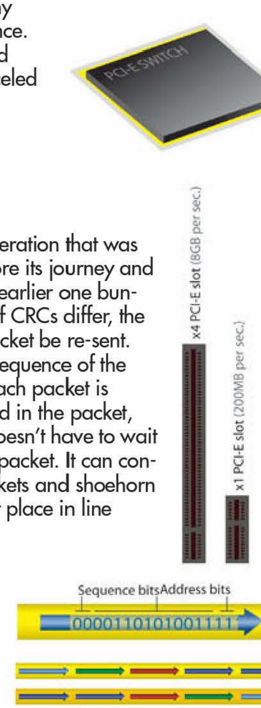
**3** In a PCI-Express bus, the South Bridge continues its relatively unheroic job of dribbling data to the pokey hard drives, USB connections, and legacy PCI cards. But now the South Bridge feeds packets to components, such as video cards, that are data speed freaks. It does so by using dedicated serial circuits for each component, simultaneous back-and-forth transmission, and parallel routes for its serial signals.

**4** The chip set sends packets serially over two lines. Another pair of lines is responsible for packets going in the opposite directions. Taken together, the two pairs are called a **lane**. One of the lines in each pair carries the original signal. The other line carries a negative image of the signal; each 0 becomes a 1 and each 1 becomes a 0. The lines are laid out so that any electrical noise, or static, that affects one line should also affect the other.




**5** When packets reach their destination, the receiver restores the negative packet to its positive version. That same operation reverses the values of any junk signals introduced by electrical interference. The bus combines the two paired packets, and any interference in the original packet is canceled by its negative image in the matching packet.

**6** It also performs the same CRC operation that was performed on the packet before its journey and compares its result to the earlier one bundled into the packet. If CRCs differ, the bus orders the packet be re-sent. Because the sequence of the data in each packet is included in the packet, the bus doesn't have to wait for the corrected packet. It can continue to accept other packets and shoehorn the corrected data into its proper place in line when it arrives.




**7** After subtracting the overhead for packet packaging, the basic PCI-Express slot has a top bandwidth of 250 megabytes a second. But PCI-E is **scalable**. Devoting two or more lanes to send data to and from a single component—called **channel bonding**—increases the bandwidth for each lane added to the channel. PCI-E transfers data at 250MB a second in each direction per lane. With a maximum of 32 lanes, PCI-E allows for a total combined transfer rate of 8GB a second in each direction. That gives a single channel nearly twice the bandwidth of the older PCI and an eight lane slot a data rate comparable to the fastest version of AGP. You can identify the expansion slots with the increased bandwidth by comparing the slots' lengths. The basic PCI-E slot is about 24.5mm long. Each 13.5mm added to other slots represents another 250MB added to their bandwidth.

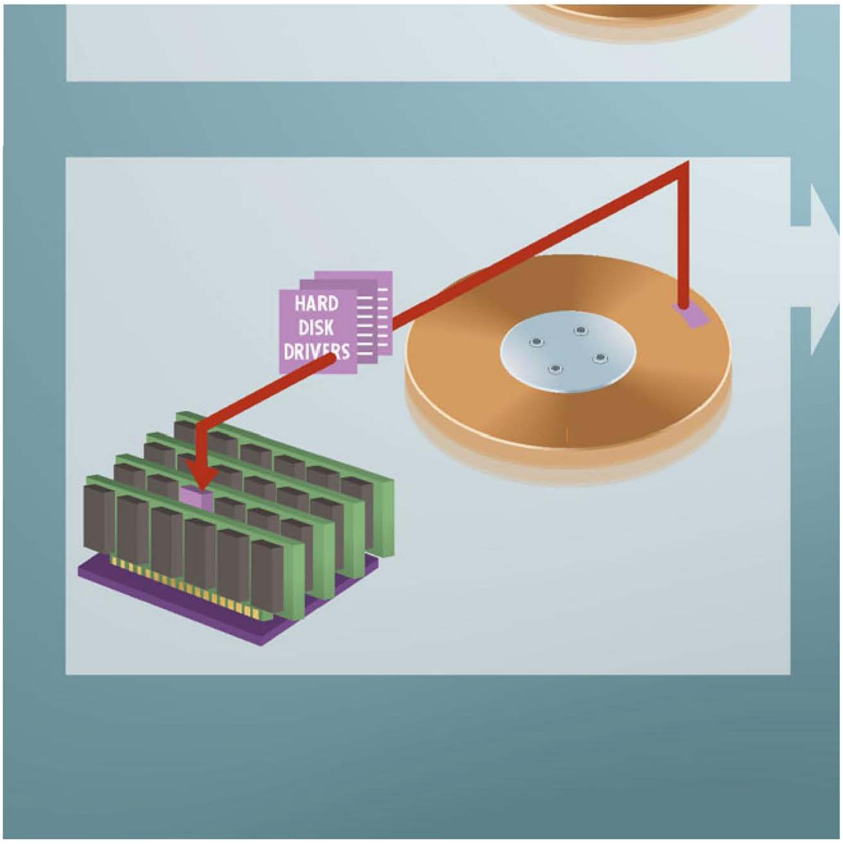
### Goodbye to the Party Line

In the older PCI bus, all the devices share the same parallel circuits and receive the same data. The data includes an identifier that says which device the signals are destined for. All other devices simply ignore them. But like telephone users on a party line, the components can't receive data while some other device monopolizes the connection. The links in PCI-E are **point-to-point**. The South Bridge uses a **crossbar switch** to route incoming signals from one point to another down circuit lines dedicated to specific components. Data goes to several components at the same time. It's like talking on a private, single-line phone.

CHAPTER

3

# How a PC Comes Alive



A personal computer can't do anything useful unless it's running an **operating system**—a basic type of software, such as Microsoft Windows, that acts as a supervisor for all the applications, games, or other programs you use. The operating system sets the rules for using memory, drives, and other parts of the computer. But before a PC can run an operating system, it needs some way to load the operating system from disk to **random access memory (RAM)**. The way to do this is with the bootstrap—a small amount of code that's a permanent part of the PC, or simply to boot.

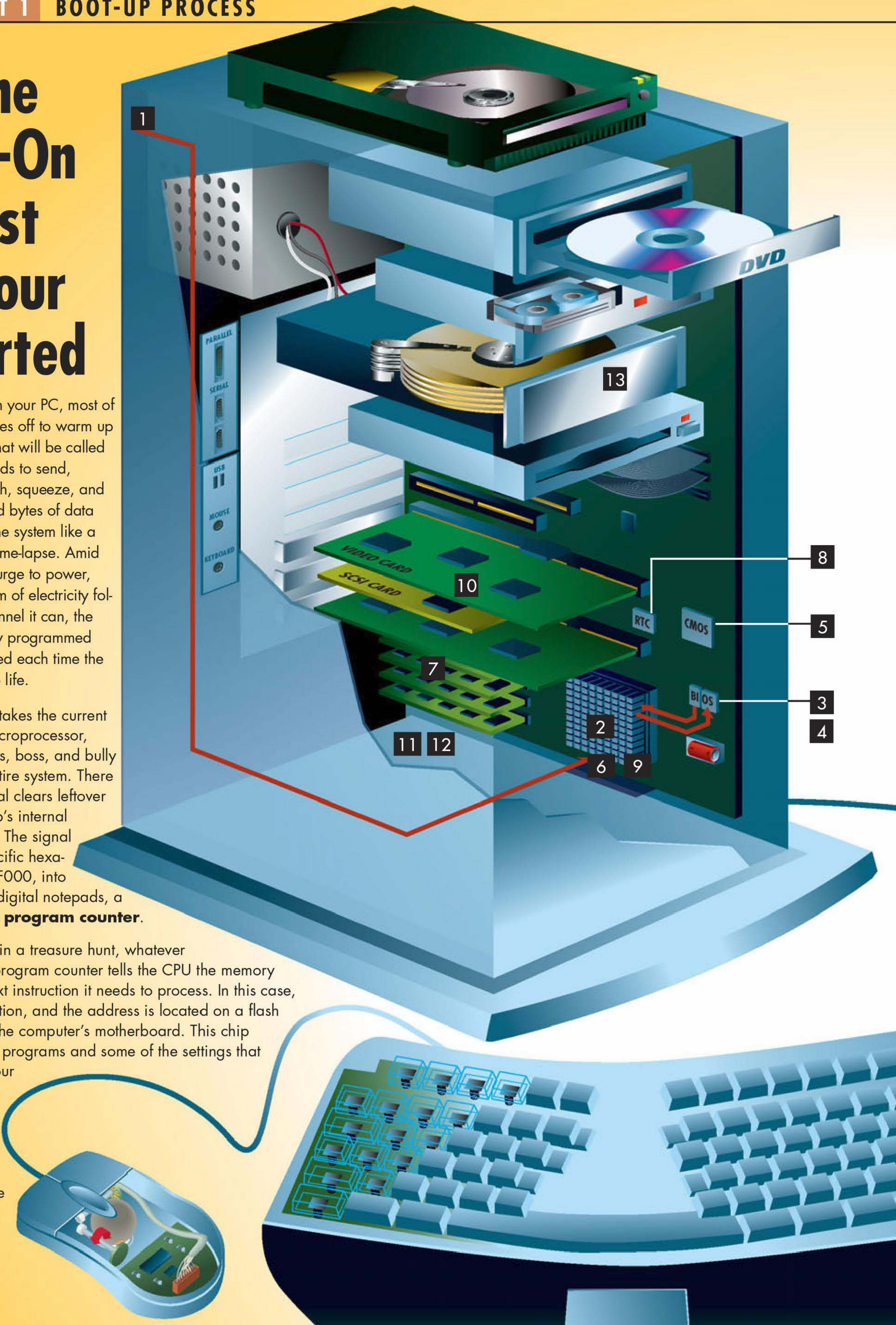
The bootstrap is aptly named because it lets the PC do something entirely on its own, without any outside operating system. Of course, the boot operation doesn't do much. In fact, it has only two functions: one is to run a **POST**, or **power-on self-test**, described on the next page of this chapter, and the other is to search drives for an operating system. When these functions are complete, the boot operation launches the process of reading the system files and copying them to random access memory.

Why do PCs use such a roundabout arrangement? Why not simply make the operating system a part of the PC? A few low-end or specialized computers do this. Early computers used primarily for playing games, such as the Atari 400 and 800, and the more recent palm-sized PCs, have a permanent operating system. But in most cases, the operating system is loaded from hard disk for two reasons.

Upgrading the operating system is simpler when loading from a disc, be it a CD, as in Windows XP, or DVD as is the case with Windows Vista. When a company such as Microsoft—which made MS-DOS and now makes Windows, the most commonly used PC operating system—wants to significantly revamp its OS, usually by adding new features or fixing serious bugs, it can simply issue a disc. Clearly, it's cheaper for Microsoft to distribute an operating system on disc than to design a microchip that contains the operating system. And it's easier for computer users to install a new operating system from disc than it is to swap chips.

The other reason for loading an operating system from disc is that it gives users a choice of operating systems. Although most PCs based on microprocessors built by Intel use Windows, there are alternative operating systems, such Linux. In some PC setups, you can even choose which of the operating systems to use each time you turn on your computer. We'll use Windows here because it's the most widely used operating system in the world.

# How the Power-On Self-Test Gets Your PC Started

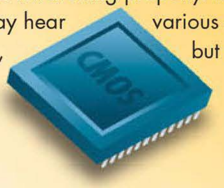



**1** When you turn on your PC, most of the electricity rushes off to warm up the components that will be called on in a few seconds to send, receive, slice, hash, squeeze, and memorize bits and bytes of data rushing through the system like a busy terminal in time-lapse. Amid this tremendous surge to power, one narrow stream of electricity follows the only channel it can, the same permanently programmed path it has followed each time the computer came to life.

**2** The familiar path takes the current to the **CPU**, or microprocessor, which is the brains, boss, and bully enforcer of the entire system. There the electrical signal clears leftover data from the chip's internal memory registers. The signal also places a specific hexadecimal number, F000, into one of the CPU's digital notepads, a register called the **program counter**.

**3** Like the next clue in a treasure hunt, whatever number is in the program counter tells the CPU the memory address of the next instruction it needs to process. In this case, it's the first instruction, and the address is located on a flash memory chip on the computer's motherboard. This chip holds a few small programs and some of the settings that determine how your computer works. All together, they're called the **BIOS** because they are fundamental to the computer's **basic input/output system**.

**4** The first trickle of electricity has done its job. Now the BIOS takes over the task of awakening the computer's components, giving them a pop quiz called the **power-on self-test (POST)** to make sure the necessary parts of the computer are present and functioning properly. It's while the POST is being administered that you may hear various churnings from your drives and see some LEDs flash, but the screen, at first, remains black.

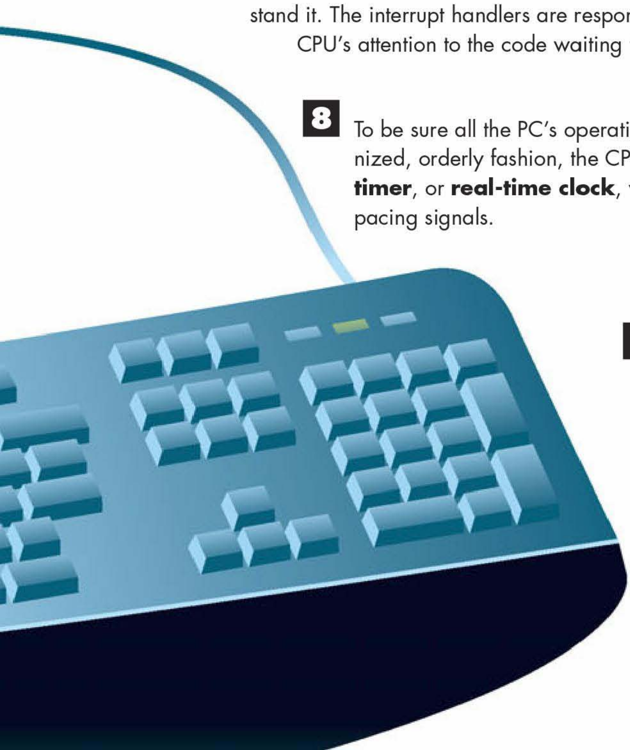



**5** The BIOS first checks a small, 64-byte chunk of RAM located on a **complementary metal oxide semiconductor(CMOS)** chip that is kept alive by a battery even when the computer is off. The CMOS contains the official record of which components are installed in your system. As the BIOS continues the POST, it uses that information as a check against the response it receives.

**6** Before inspecting other parts of the components, the BIOS and CPU check to make sure they're working right. The BIOS instructs the CPU to read code stored at various locations and compares what it finds to identical records stored permanently in the BIOS chip set.

**7** The BIOS loads into memory the **device drivers** and **interrupt handlers** from the basic hardware in the system, such as the keyboard, mouse, hard drive, and floppy drive. Whenever you press a key, the keyboard generates a code specific to that key, and the device driver translates the code as needed for the CPU to understand it. The interrupt handlers are responsible for bringing the CPU's attention to the code waiting for the microprocessor.

**8** To be sure all the PC's operations function in a synchronized, orderly fashion, the CPU also checks the system's **timer**, or **real-time clock**, which is responsible for pacing signals.

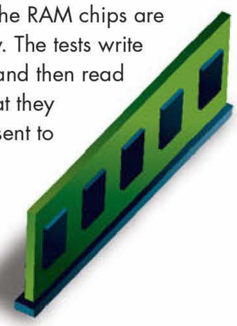


**9** The BIOS has the CPU send signals over the **system bus** to be sure all of the basic components are functioning. The bus includes the electrical circuits printed on and into the motherboard, connecting all the components with each other.

**10** The POST tests the memory contained on the display adapter and the video signals that control the display. It then makes the adapter's BIOS code a part of the system's overall BIOS and memory configuration. This is the first point you'll see something appear on your PC's monitor.

**11** The BIOS checks to see if it's engaged in a **cold boot**, meaning the computer had been turned off, or if it's a **warm boot**, or **reboot**, by checking the value at memory address 0000:0472. If it finds the number 1234, the BIOS knows this is a reboot and skips the rest of the POST.

**12** For a cold boot, the BIOS runs a series of tests to ensure that the RAM chips are functioning properly. The tests write data to each chip, and then read it and compare what they read with the data sent to the chips in the first place. At this point on some PCs, you'll see a running account of the amount of memory that's been checked on the monitor.

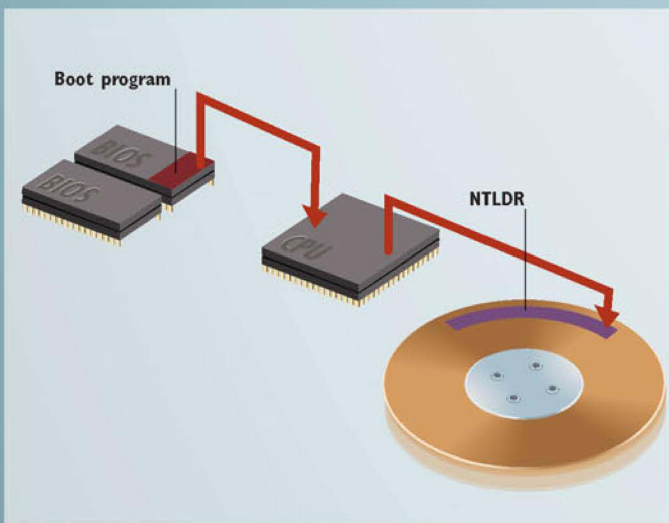


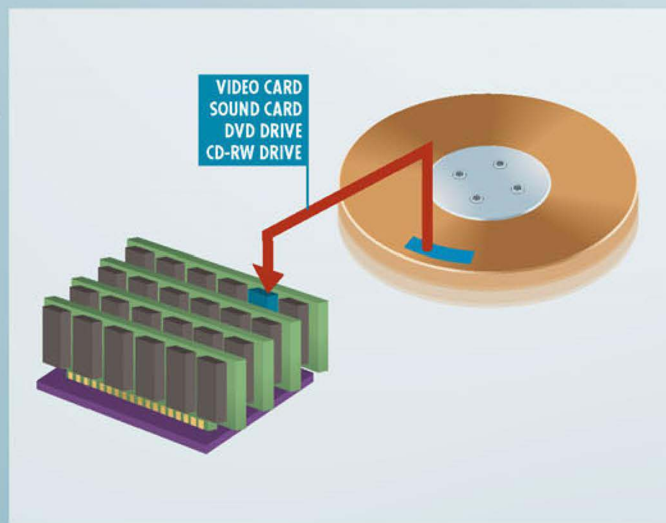
**13** The POST sends signals over specific paths on the bus to the internal floppy, optical, and hard disk drives, and listens for a response to determine which drives are available. That ends the POST and the BIOS transfers control of the PC to the operating system on the hard disk, a process called the **boot**.

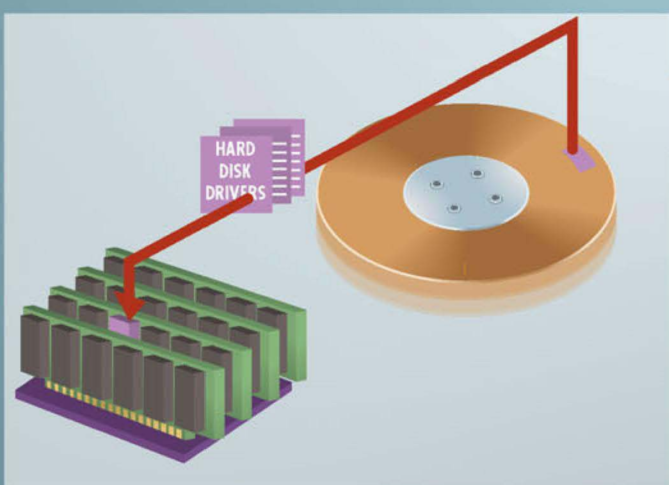
# How a Disk Boot Wakes Up Your PC

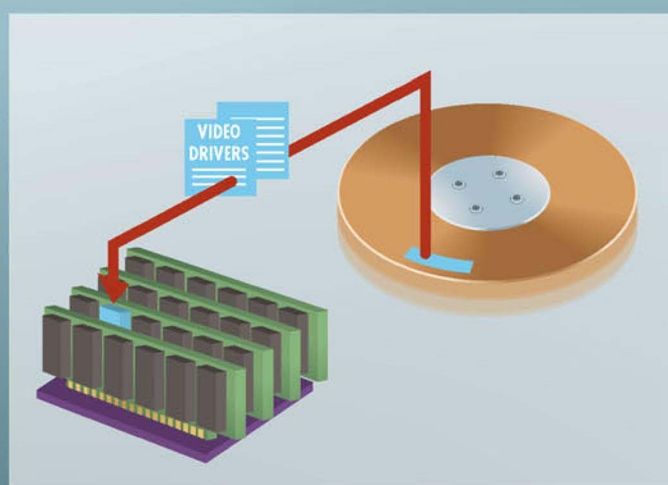
**1** After conducting a POST check of all the hardware components of a PC, the boot program contained on the computer's ROM BIOS tells the processor to execute a program contained in the C: hard drive's **boot sector** (or the CD-ROM or floppy disk if there's no boot sector of the hard drive). That code, **NTLDR (NT LoaDeR)**, tells the processor where to find more code on the drive.

**2** On a Windows XP system, that code is **NTDETECT.COM**, which displays the Windows XP **splash screen** before making a list of all the system's hardware it recognizes. NTDETECT displays the list onscreen and passes it along to the Windows **Registry**, where other programs have access to the information.



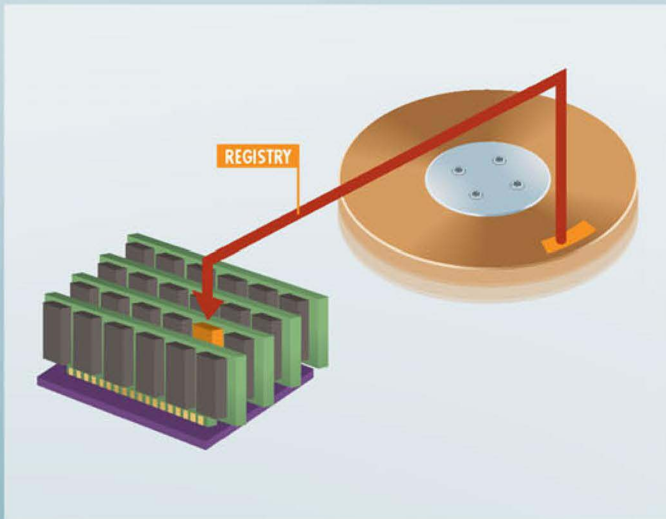

**5** The next step is to prepare the computer to hold the gigabytes of new files that constantly grow with a Windows installation. Windows XP loads support for hard drives, maintains disk partitions, and officially mounts the drives for use.

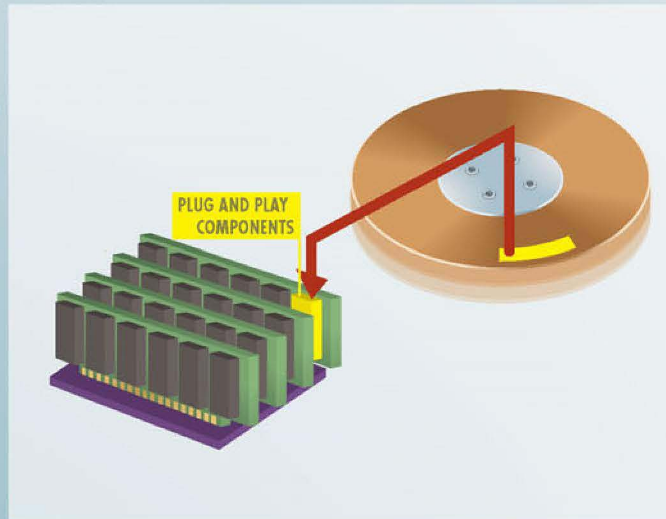
**6** With its drive support loaded, Windows XP then loads special video drivers for such components as a **PCI-Express** video card or MPEG software support for video.

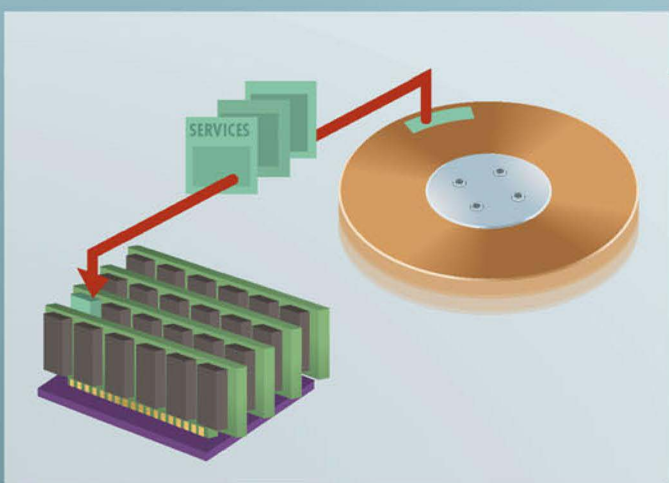
A computer is only a collection of millions of possibilities until it has an **operating system (OS)**. The collection of rules and instructions in an operating system make the computer a Windows XP or Vista machine, a Linux computer, a Windows 98 PC, or even an MS-DOS computer. Whatever flavor of OS goes into a computer takes over the PC like a binary body snatcher. For this explanation, we're going to assume the operating system taking hold in the computer drive's boot sector is Windows XP.

**3** The Registry takes some of the chores of booting the system off NTLDR's back. The Registry loads several **low-level** programs into memory. These are programs working at the most basic machine level to control the hardware. This is when those proletariat programs help Windows XP expand the machine's artificial consciousness by loading still other programs that Windows quickly assimilates as parts of an operating system.

**4** With enough files located to handle basic hardware operations, Windows begins another survey of components, this time loading Plug and Play's **enumerator**. (See p. **42** for more on PnP.) This process also loads drivers for the PCI bus and those of the older ISA bus, should the computer have one.



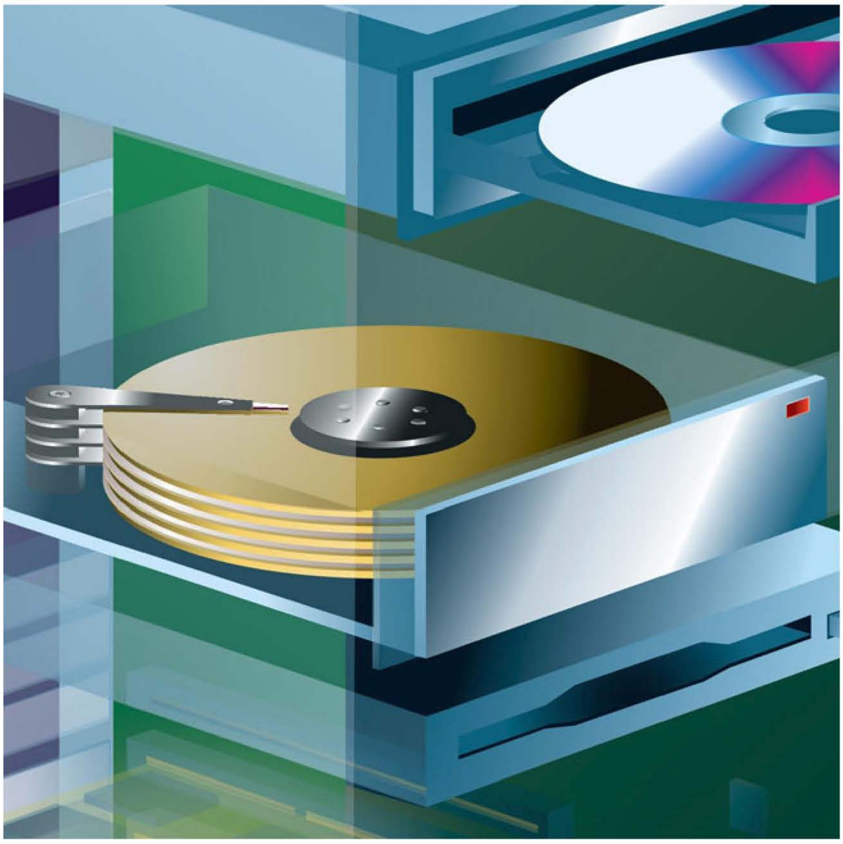

**7** Next to join the mixture are **services**. These are essential services, such as disk defragging and partition management. Other services, such as remote access, might have no appeal for some PC users.


**8** When the services have all been installed and begin doing their jobs, you finally see the Windows logo onscreen. The operating system is ready to get to work.

CHAPTER

4

# How an Operating System Controls Hardware



**OPERATING** systems were originally developed to handle one of the most complex input/output operations: communicating with a variety of disk drives. This is evidenced by the names given to early operating systems, which often contained the acronym DOS, for *disk operating system*. Eventually, the operating system quickly evolved into an all-encompassing bridge between your PC and the software you run on it.

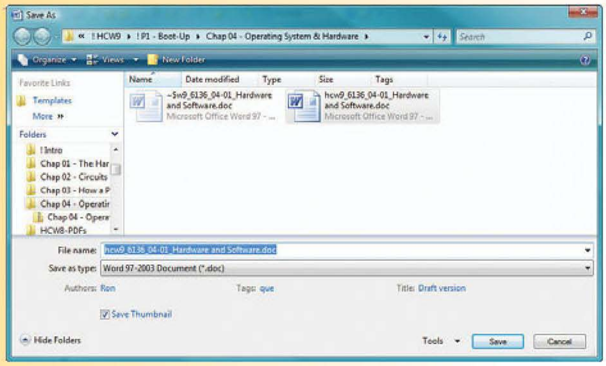
Without an operating system, such as Windows, each programmer would have to invent from scratch the way a program displays text or graphics onscreen, how it sends data to the printer, how it reads or writes disk files, and how it implements other functions that mesh software with hardware. An operating system, however, is more than a way to make life easier for programmers.

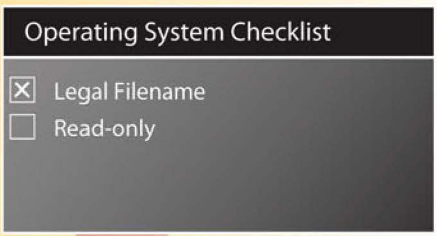
An operating system creates a common platform for all the software you use. Without an operating system, you might not be able to save files created by two different programs to the same disk because each might have its own storage format. An operating system also gives you a tool for all the tasks you want to perform outside an application program: deleting and copying files to disk, printing, and running a collection of commands in a batch file.

The operating system does not work alone. It depends not only on the cooperation of other programs, but also on meshing smoothly with the BIOS and software drivers. The *BIOS*—or basic input/output system—is made of code contained on chips in a PC. It acts as the intermediary among the hardware, processor, and operating systems. Device drivers are like a specialized BIOS. Drivers translate commands from the operating system and BIOS into instructions for a specific piece of hardware, such as a printer, scanner, or DVD-ROM drive. When some parts of the operating system are loaded from disk, they are added to the BIOS and then joined by device drivers, and all of them carry out routine hardware functions. The operating system is really composed of all three of these components, plus scores of other programs, common code, and data files.

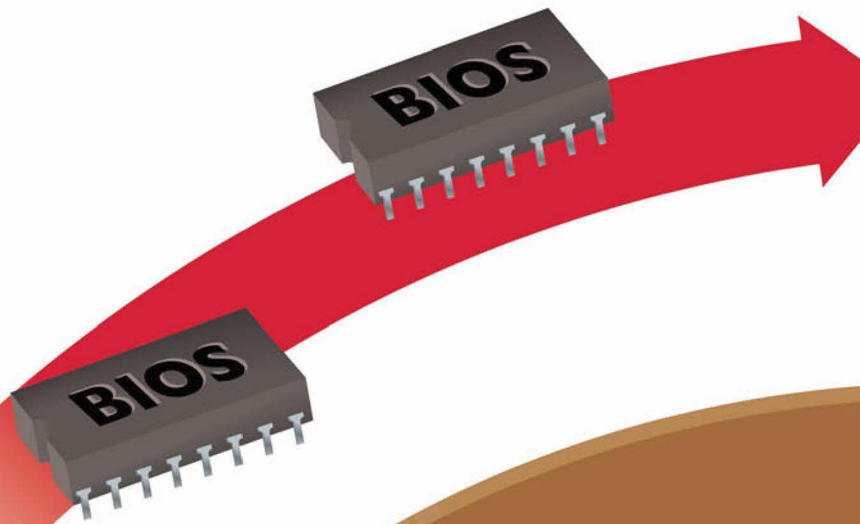
Together, the BIOS, device drivers, and Windows perform so many functions that it's impossible to depict their complexity with a couple of pages of illustrations. Here we'll show how the operating system and Plug and Play work, how a PC's software and hardware components work together, and how hardware interrupts whatever software is doing to get some attention from the processor.

# How Hardware and Software Work Together

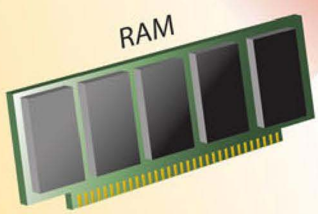



**1** In the early days of personal computers, only a few dozen hard drives existed that could be used in a PC. But the drives had mechanical and electronic differences; One might've had twice the capacity, more speed, or used platters that were an odd size compared to another. Because there were so few drives, IBM and the companies that cloned the original PC included instructions in the computer's **BIOS** chip for every hard drive that someone was likely to install. This, of course, was not a long-term solution. As the number of hard drives evolved as rapidly as fruit flies, their instructions outgrew the capacity of the BIOS. The solution was **device drivers**, coded instructions for a specific hardware peripheral contained in a file and copied to the computer when that device is installed. The device can also be a mouse, printer, keyboard, or any other equipment that needs custom handling, and each drive is tailored for a specific brand, size, and design of whatever hardware it's written for. The device driver has an important role when, for example, you save a document.



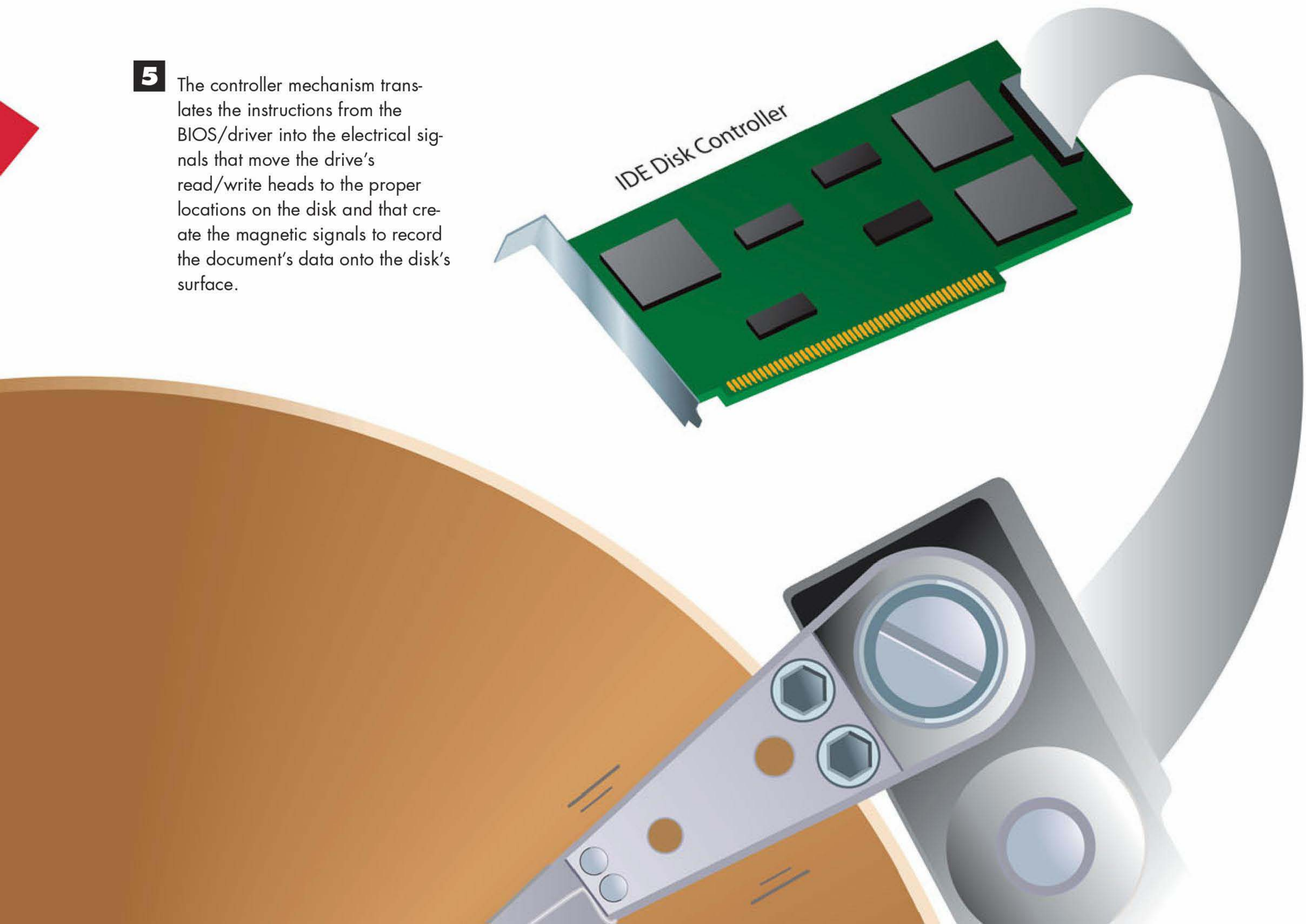

**4** On older systems, if the disk drive is one for which the BIOS maintains a specifically tailored set of instructions, the BIOS itself sends the instructions along with the data to the mechanism that controls the disk drive. On new systems, the BIOS is more likely to retrieve the instructions from the device driver.



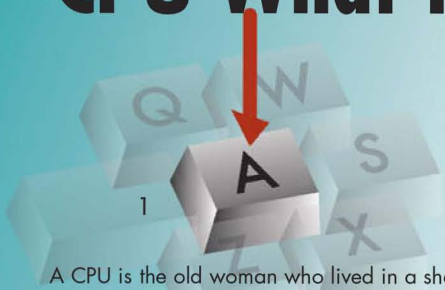
**2** When you tap a key or click an icon to save a word processing file, the software, which contains virtually no code for dealing directly with a disk drive, hands off the command to the **operating system**, usually Windows. The operating system checks to be sure that there are no problems with the command to save the data. For example, it makes sure that the filename is a legal one—that it does not contain any forbidden characters such as * and ?—and that you're not trying to save a file with the same name as a file marked read-only.

**3** If everything is on the up and up, the operating system checks whether operation requires a device driver; these days it almost always does. Most drivers are loaded when the operating system is loaded, and become seamless extensions of the BIOS. For this save operation, Windows simply turns over the nitty-gritty job of saving the document to the BIOS and the driver.

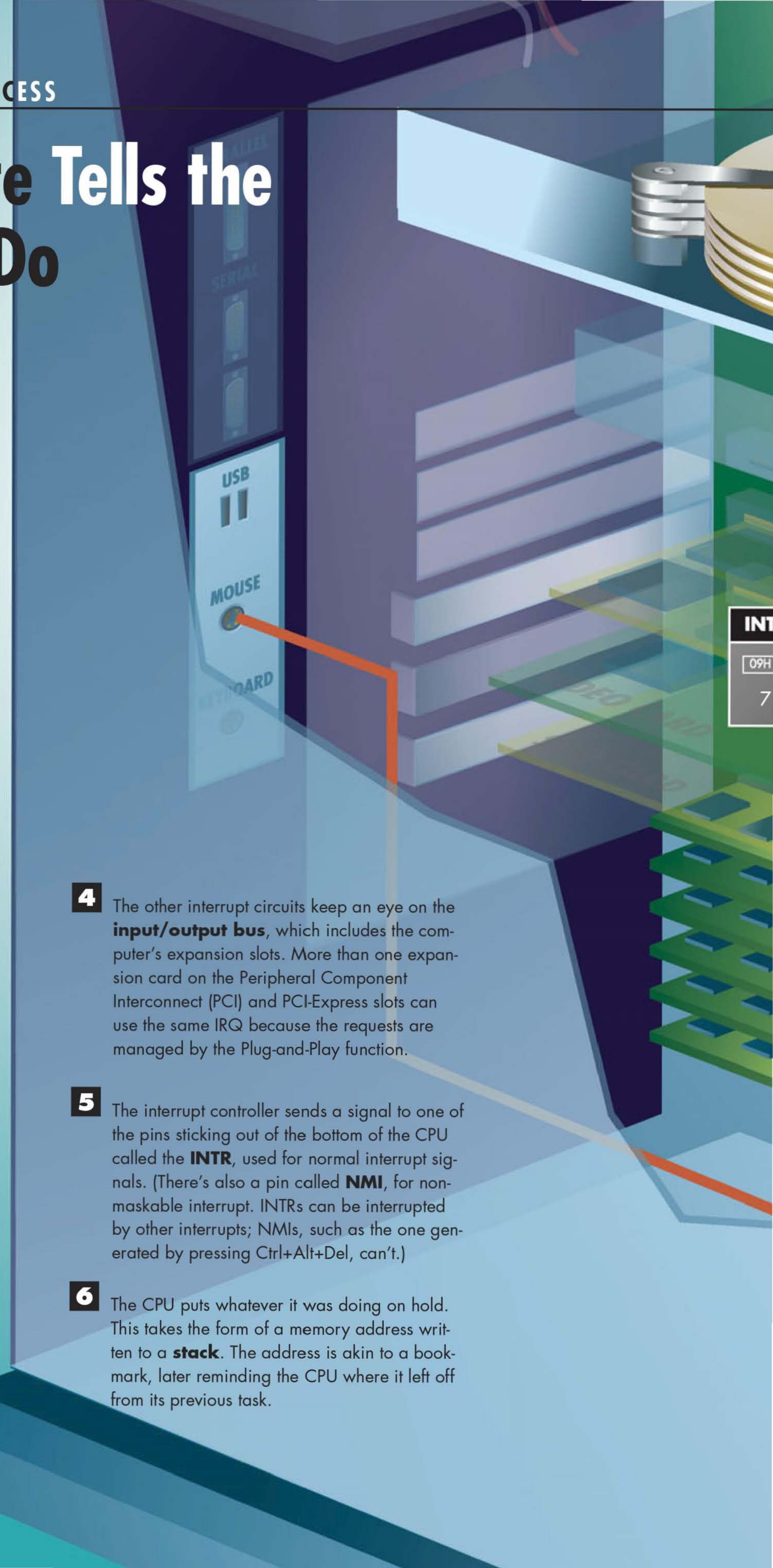
**5** The controller mechanism translates the instructions from the BIOS/driver into the electrical signals that move the drive's read/write heads to the proper locations on the disk and that create the magnetic signals to record the document's data onto the disk's surface.



# How Hardware Tells the CPU What to Do




A CPU is the old woman who lived in a shoe of computing. This mother didn't know what to do because she didn't devise a method of responding to the demands of all her children. She could have borrowed one of two methods from computing: polling and interrupts. With polling, she would have gone from child to child endlessly, asking if she could do anything for them. If one needed something, the other children would have to wait until that task was completed, even if they had an urgent need to use the bathroom. Not very efficient. With interrupts, Mother would go about her work until a child tugs at her apron, asking for something. The interruption gets her immediate attention, at least until another child needs something. In computers, as in shoe houses, interrupts are the preferred method of handling requests from software and hardware.




**1** When you press the A key, an electrical signal travels along a circuit, which serves to identify what key you pressed, to the **keyboard controller**.

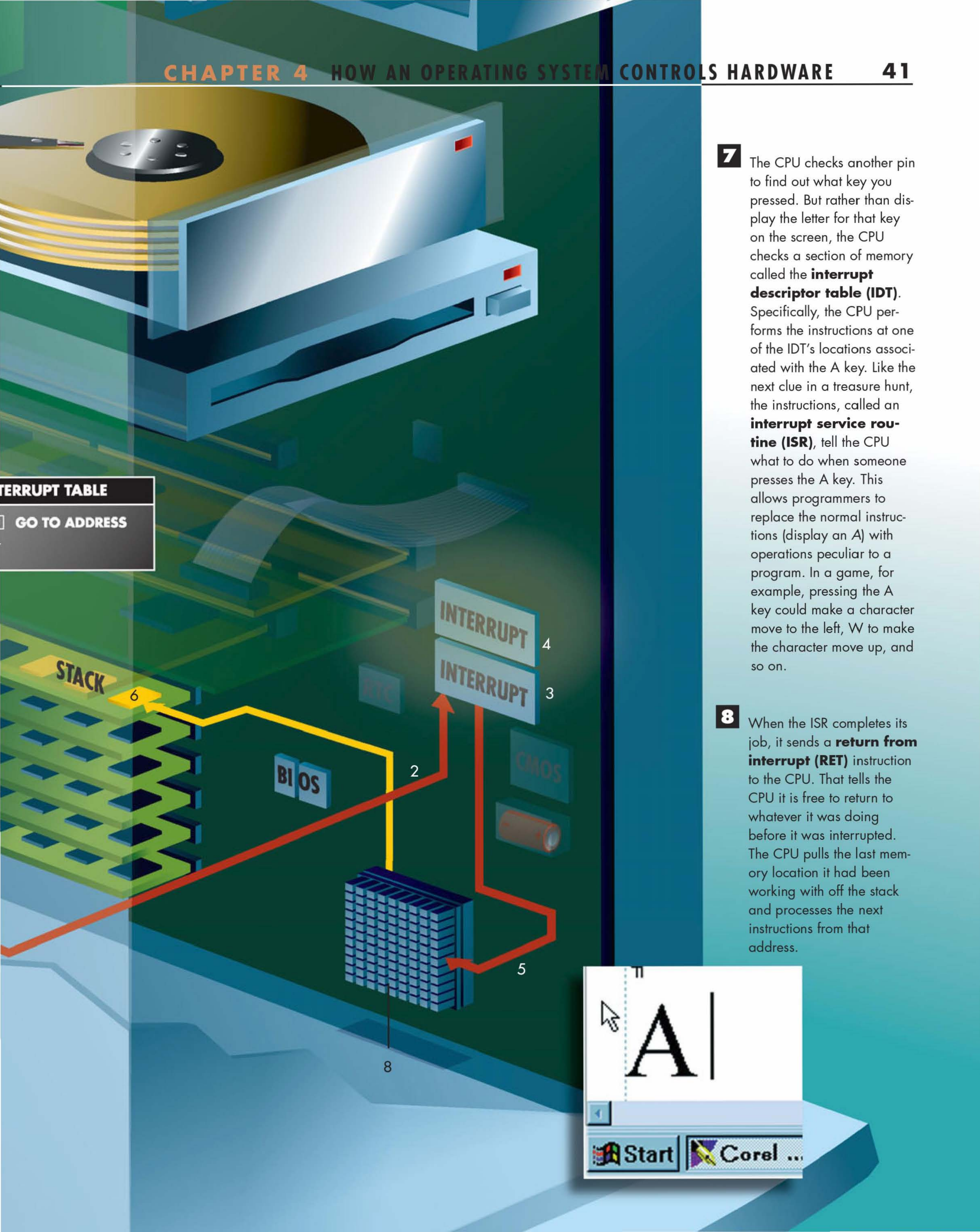
**2** The keyboard interrupt arrives on one of 16 **interrupt request (IRQ) lines**. Seven of the IRQs monitor specific components, such as the keyboard controller.

**3** The controller relays a signal to a subsystem called the **interrupt controller,** which acts as chief-of-staff, running interference between the CPU and the 256 possible kinds of interrupts that clamor for the CPU's attention.

**4** The other interrupt circuits keep an eye on the **input/output bus**, which includes the computer's expansion slots. More than one expansion card on the Peripheral Component Interconnect (PCI) and PCI-Express slots can use the same IRQ because the requests are managed by the Plug-and-Play function.

**5** The interrupt controller sends a signal to one of the pins sticking out of the bottom of the CPU called the **INTR**, used for normal interrupt signals. (There's also a pin called **NMI**, for nonmaskable interrupt. INTRs can be interrupted by other interrupts; NMIs, such as the one generated by pressing Ctrl+Alt+Del, can't.)

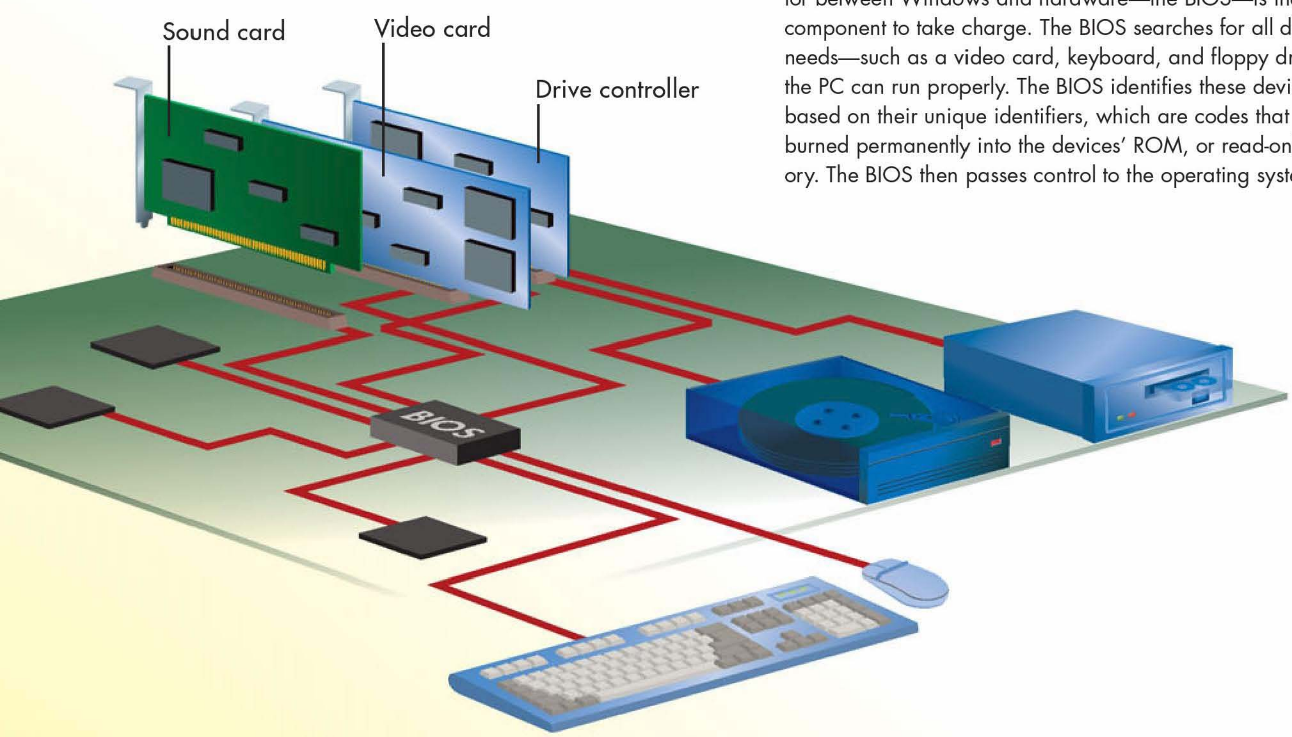
**6** The CPU puts whatever it was doing on hold. This takes the form of a memory address written to a **stack**. The address is akin to a bookmark, later reminding the CPU where it left off from its previous task.




**7** The CPU checks another pin to find out what key you pressed. But rather than display the letter for that key on the screen, the CPU checks a section of memory called the **interrupt descriptor table (IDT)**. Specifically, the CPU performs the instructions at one of the IDT's locations associated with the A key. Like the next clue in a treasure hunt, the instructions, called an **interrupt service routine (ISR)**, tell the CPU what to do when someone presses the A key. This allows programmers to replace the normal instructions (display an *A*) with operations peculiar to a program. In a game, for example, pressing the A key could make a character move to the left, W to make the character move up, and so on.

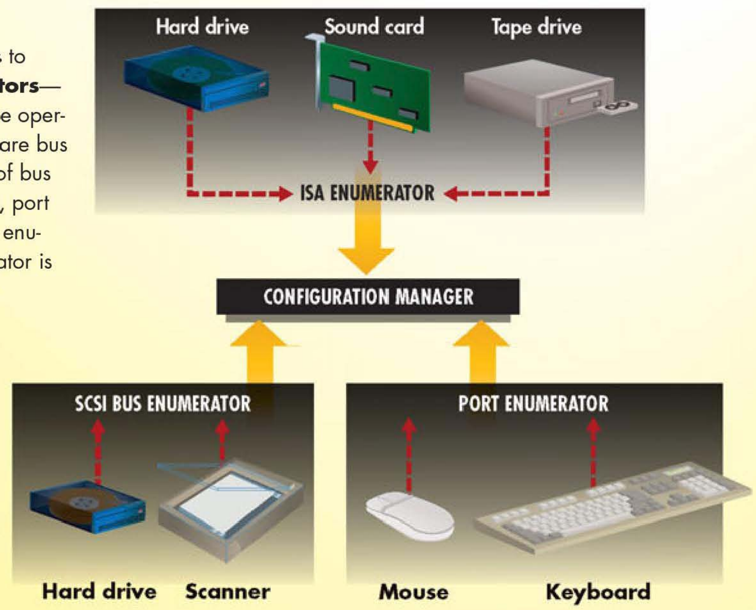
**8** When the ISR completes its job, it sends a **return from interrupt (RET)** instruction to the CPU. That tells the CPU it is free to return to whatever it was doing before it was interrupted. The CPU pulls the last memory location it had been working with off the stack and processes the next instructions from that address.

# How Plug and Play Works

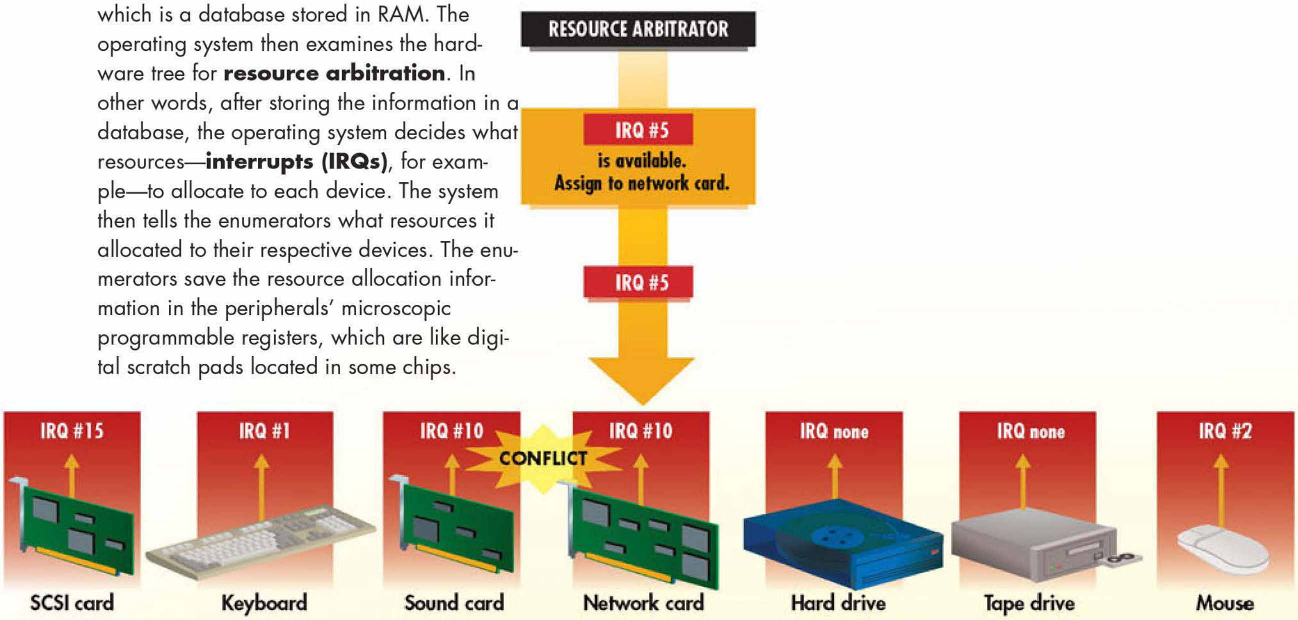



**1** When you turn on a Plug-and-Play system, the primary arbitrator between Windows and hardware—the BIOS—is the first component to take charge. The BIOS searches for all devices it needs—such as a video card, keyboard, and floppy drive—so the PC can run properly. The BIOS identifies these devices based on their unique identifiers, which are codes that are burned permanently into the devices' ROM, or read-only memory. The BIOS then passes control to the operating system.

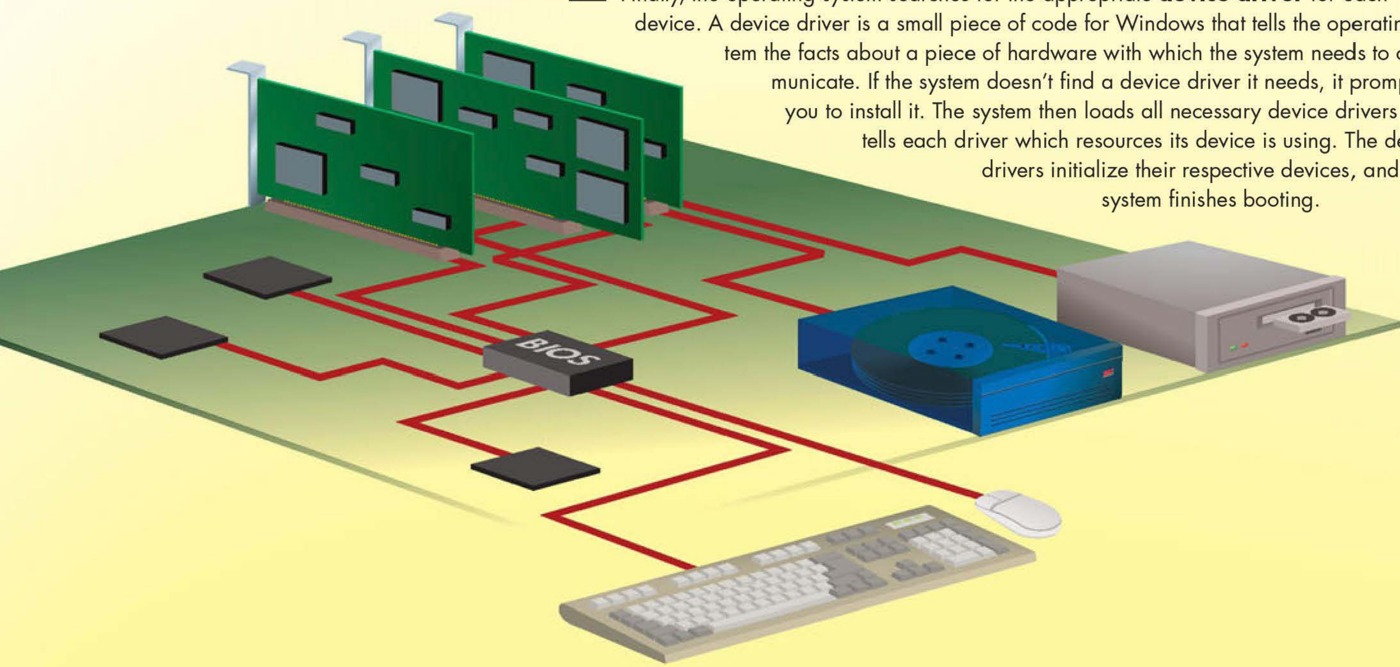
**2** The configuration manager in Windows adds to itself special device drivers called **enumerators**—programs that act as the interface between the operating system and the different devices. There are bus enumerators, enumerators for a special type of bus called SCSI (small computer system interface), port enumerators, and more. Windows asks each enumerator to identify which devices the enumerator is going to control and what resources it needs.



**3** Windows takes the information from the enumerators and stores it in the hardware **tree**, which is a database stored in RAM. The operating system then examines the hardware tree for **resource arbitration**. In other words, after storing the information in a database, the operating system decides what resources—**interrupts (IRQs)**, for example—to allocate to each device. The system then tells the enumerators what resources it allocated to their respective devices. The enumerators save the resource allocation information in the peripherals' microscopic programmable registers, which are like digital scratch pads located in some chips.

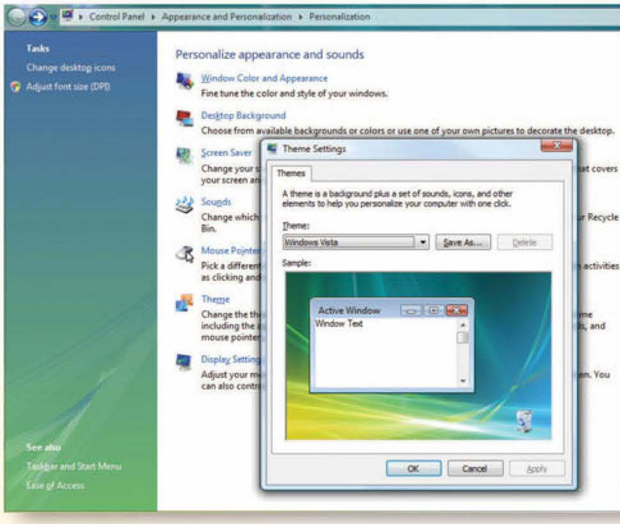



**4** Finally, the operating system searches for the appropriate **device driver** for each device. A device driver is a small piece of code for Windows that tells the operating system the facts about a piece of hardware with which the system needs to communicate. If the system doesn't find a device driver it needs, it prompts you to install it. The system then loads all necessary device drivers and tells each driver which resources its device is using. The device drivers initialize their respective devices, and the system finishes booting.



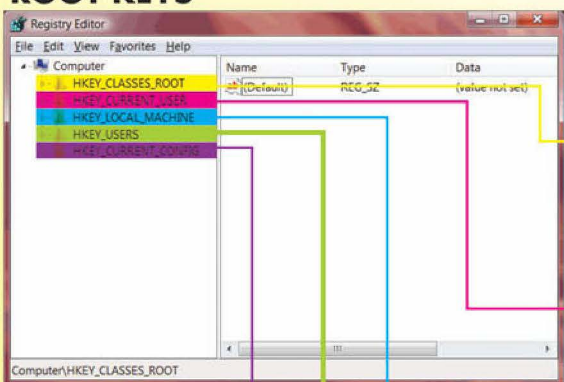
# How the Registry Knows It All

If you just want a passing idea of how complex Windows is, take a look at the Windows Registry. If you're not sure how to do that, don't even *think* the word *registry* until you learn more about it. In the Registry, you'll find 90-something percent of the filenames, paths, parameters, programs, and patches that keep the operating system running without a hitch. Here's just a hint of what you'll encounter.




**1** The Windows Registry is a hierarchical database of configuration settings that controls all aspects of how Windows looks and works. When you use a menu to change Windows in some way (for example, deciding to use the classic Start menu or the newer style Windows Start menu), you're actually making a change to the Registry. Many of the menus and dialog boxes you use to configure Windows and Windows programs are only front ends to protect you from having to stare directly into the maw of the beast.

## ROOT KEYS




**2** In some instances, you can't make changes to Windows without directly editing the Registry using a built-in Windows tool called the **Registry Editor** (Regedit if you using it from a *run* command).

**3** The Registry is organized in a hierarchical manner, with **keys** at the top, **subkeys** beneath them, and still more **values** beneath them. At the very top of the Registry are five **root keys** that each controls a different aspect of how Windows works:

**HKEY_CLASSES_ROOT (HKCR)**—The settings under this root key contain information about file types and filename extensions. The settings tell Windows how to handle files, and they control basic interface options, such as double-clicking and context menus.

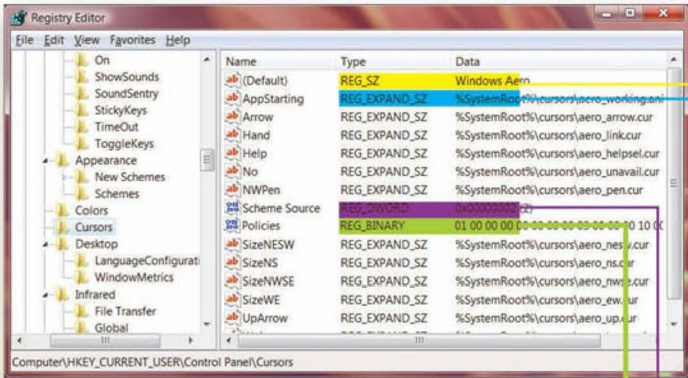
**HKEY_CURRENT_USER (HKCU)**—These control the settings of the user currently logged into Windows. They control the current user's Desktop, as well as Vista's or XP's appearance and behavior They manage network connections, connections to devices such as printers, and display personal preferences. Also included are **Security Identifiers** (SIDs) that identify users of the PC and have information about each user's rights, settings, and preferences.

**HKEY_LOCAL_MACHINE (HKLM)**—These settings track the computer, its hardware, and the operating system. They include specific details about all hardware, including the keyboard, printer ports, and storage devices. They also contain information about security, installed software, system startup, drivers, services, and the machine's specific Vista or XP configuration. They affect every user who logs onto the computer.

**HKEY_USERS (HKU)**—These keys control the user-by-user settings Windows uses to display the default desktop—the desktop that displays before anyone logs on—as well as user profiles. It contains information about every user profile on the system.

**HKEY_CURRENT_CONFIG (HKCC)**—These settings control configuration of the current hardware profile. It contains information about the hardware configuration in the same way that HKEY_CURRENT_USER contains information about the current user of the system.

## SUBKEYS




**4** Individual keys and subkeys control every aspect of how Windows works. As a way to control the computer and Windows, a key uses one and sometimes more than one value. The five primary data types of values are:

**REG_SZ (string value)**—This data type is made up of plain text and numbers and is one of the most common data types in the Registry. For example, the key DoubleClickSpeed determines the amount of time between mouse clicks that must elapse before Windows won't consider it a double-click.

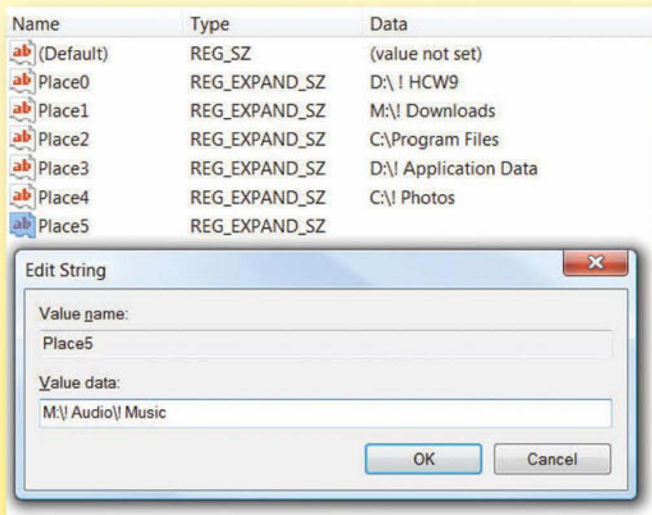
**REG_MULTI_SZ (string array value)**—This data type contains several strings of plain text and numbers. The Registry Editor enables you to edit these values, but won't let you create them.

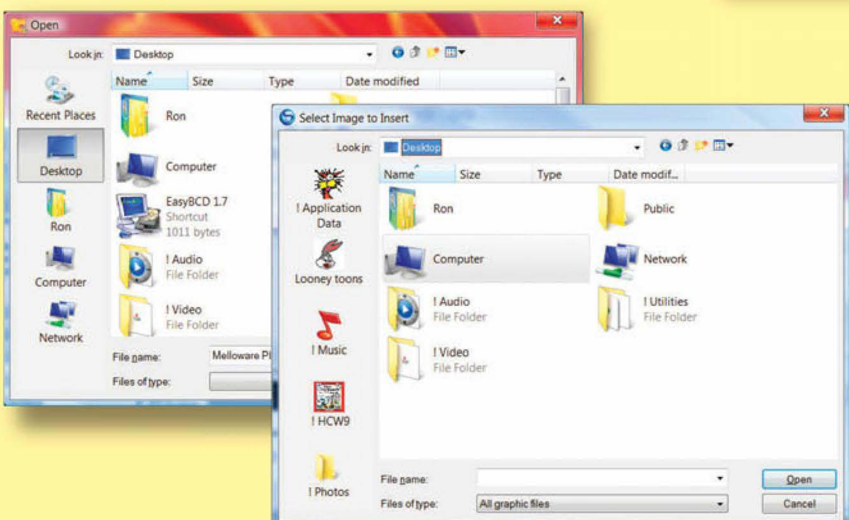
**REG_EXPAND_SZ (expanded string value)**—This data type contains variables that Windows uses to point to file locations. For example, the expanded string value used to point to the location of the Aero theme file, which gives Vista its glassy transparency, is *%SystemRoot%\resources\Themes\Aero\Aero.msstyles*.

**REG_DWORD (DWORD values)**—This data type is represented as a number. Sometimes 0 turns on the key or 1 turns off the key, although other numbers can be used as well. Although you see and edit the value as a number, the Registry itself views the number as a hexadecimal number.

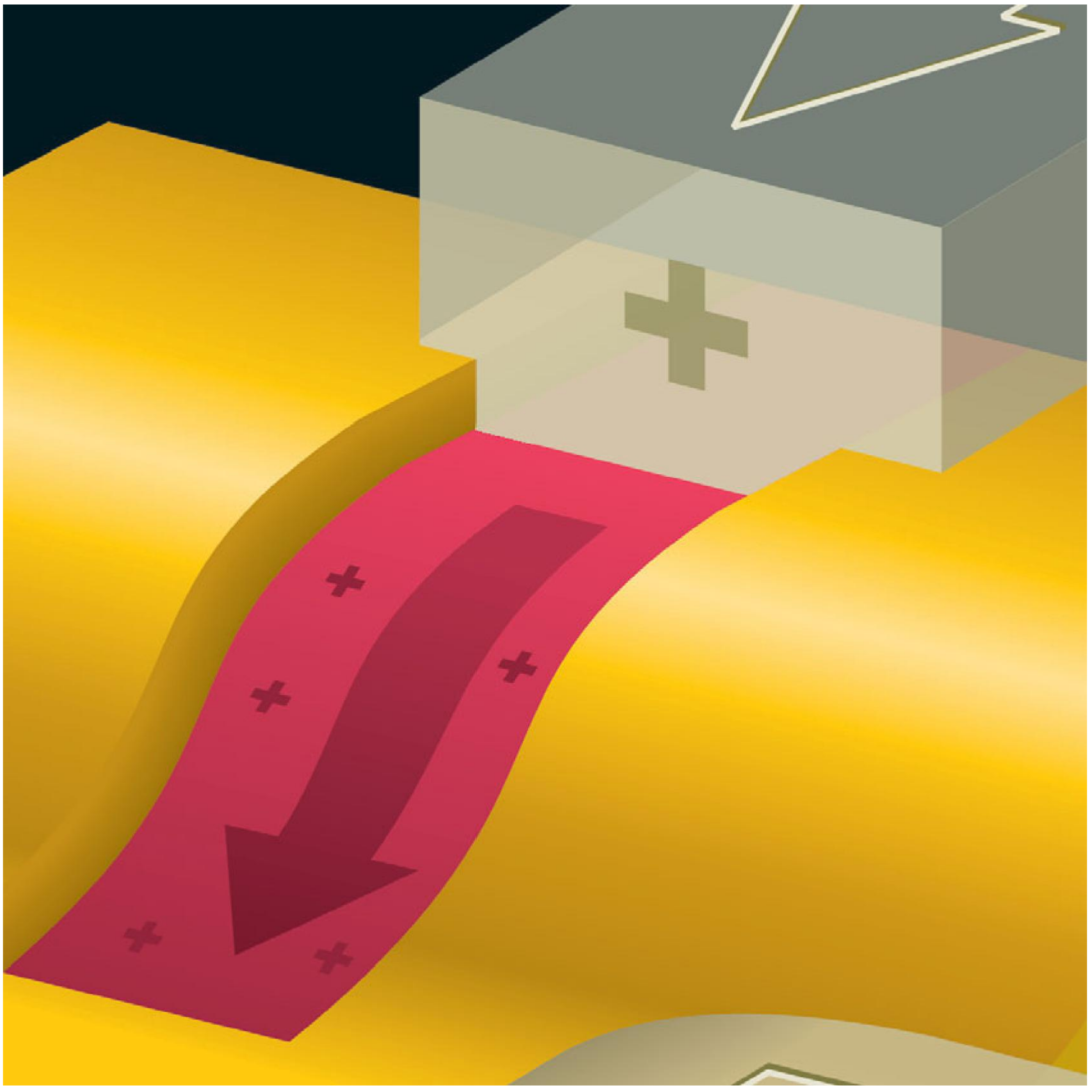
**REG_BINARY (binary values)**—This data type is made up of binary data, 0s and 1s.

**5** To change or create a Windows Vista or XP setting, you need to know the proper Registry key or subkey and the correct value for the change you want to make. For example, if you want to change the **Places**, the shortcuts to often used folders that are displayed to the left of Open and Save dialog boxes, you must find a Registry key at *HKEY_USERS\S-1-5-21-2505112949-3015503851-117469413-1000\Software\Microsoft\Windows\CurrentVersion\Policies\Comdlg32*. There, through the use of a right-click, you create a new subkey, which I've named PlacesBar. Another right click there allows you to add subkeys for Places0 through Places4. You provide the values for them in the form of the directory paths you want displayed in the dialog boxes.




**6** Here is a dialog box before and after the creation of PlacesBar using Regedit. But remember, editing the Registry is a tricky business that could leave you unable to use Windows. A safer approach is to look on the Internet for programs others have devised that let you change Registry settings safely. To change the folders in PlacesBar, you can download the free program PlacesBar Editor from http://melloware.com/.



**1350 B.C.**
The Chinese use the first decimal. The floating-point microprocessor, an American invention that's dependent upon using decimals, follows 3,330 years later.

**945 B.C.**
The churchman Gerbert, who later becomes Pope Sylvester II, introduces the abacus and Hindu-Arabic math to Europe. But this new method for writing numbers does not catch on.

**1623**
Wilhelm Schickardt invents the Calculating Clock, the first mechanical calculator, based on the idea of Napier's Bones—rods used as mechanical aids to calculation and envisioned by John Napier in 1614.

**1671**
Gottfried Leibniz introduces the Step Reckoner, a device that can multiply, divide, and evaluate square roots.

**1679**
Gottfried Leibniz introduces binary arithmetic—a fundamental discovery showing that every number can be represented by the symbols 0 and 1 only.

**1820**
Charles Xavier Thomas de Colmar develops the "arithometer," the first practical and reliable device capable of the four basic arithmetic functions: addition, subtraction, multiplication, and division.

**1823**
Baron Jons Jakob Berzelius isolates silicon (Si), later to become the basic constituent of microchips.

**1854**
Augustus DeMorgan, in conjunction with Boole, formalizes a set of logical operations now known as DeMorgan transformations.

**1886**
Heinrich Rudolf Hertz, of megahertz (MHz) fame, proves that electricity is transmitted at the speed of light.

PART

# 2

# How Microchips are the PC's Brain

CHAPTERS



**1903**
Nikola Tesla, a Yugoslavian scientist and inventor, patents electrical logic circuits called "gates" or "switches."

**1904**
John Ambrose Fleming experiments with Edison's diode vacuum tube (an invention Edison did not pursue), developing the first practical radio tubes. The vacuum tube's application to computers is not exploited until the late 1930s.

**1926**
First patent for semiconductor transistor. The transistor allows electrical currents to flow through a computer, allowing data to be passed through the machine.

**1939**
John Atanasoff conceptualizes the ABC—the firstprototype machine to use vacuum tubes. It isthe first electronic digital computer.

**1943**
First electronic general-purpose computer, the ENIAC, has 19,000 vacuum tubes, 1,500 relays, and consumes almost 200 Kilowatts of electricity.

**1946**
After leaving the University of Pennsylvania due to disagreements about patent ownership, J. Presper Eckert and John Mauchly—the two men who headed the ENIAC project—launch the first commercial computer company, Electronic Control Company. They begin work on the UNIVAC (Universal Automatic Computer) for the U.S. Census Bureau.

**1947**
Jay Forrester extends the life of a vacuum tube from 500 hours to 500,000, improving the reliability of the vacuum-based computers of the era.

**1948**
John Bardeen, Walter Brattain, and William Shockley of Bell Labs file for a patent on the first transistor.

*"But what ... is it good for?"*

—Engineer at the Advanced Computing Systems Division of IBM, 1968, commenting on the microchip

**THOMAS** Edison in 1883 noticed that electrical current flowing through a light bulb's filament could make the wire so hot that electrons boiled off, sailing through the vacuum inside the bulb to a metal plate that had a positive charge. Because Edison didn't see any way the phenomenon would help him perfect the light bulb, he only made a notation of the effect, which he named after himself. The effect sat on the shelf until 1904, when a former Edison employee, inventor John Fleming, went to work for the Marconi Radio Company. For his first assignment, finding a better way to receive distant radio signals, Fleming began experimenting with the Edison effect. He discovered that radio waves passing through an airless tube created a varying direct current, which could be used with headphones to reproduce the sound carried by the waves. Fleming named it the oscillation valve and applied for a patent. Marconi, though, chose another, less expensive technology: a crystal wave detector.

The discoveries sat on the shelf until radio pioneer Lee DeForest read about Fleming's valve and built one himself. The valve he created in 1906 had something new: a grid made of nickel wire placed between the filament and the plate. Applying even a small electrical charge to the grid disrupted the flow of electrons from the filament to the plate. It was the beginning of the vacuum tube, which essentially let a small amount of electrical current control a much larger flow of current.

If you're not at least old enough to be part of the baby boom generation, you might never have seen more than one type of vacuum tube—the cathode ray tube (CRT) that displays images on desktop PC monitors and the ordinary TV screen. Except for CRTs—and in the sound systems of audiophiles who swear vacuum tube amplifiers are better than transistorized amps—vacuum tubes are rarely used in modern electronics. It isn't the amplifying capabilities of the vacuum tube that have made it one of the seminal discoveries of science; It is the vacuum tube's capability to act as a switch. When a small amount of current was applied to the grid, it turned off a much stronger current. Turn off the electricity going to the grid, and the larger current is switched back on. On, off. Off, on. Simple.

**1950**
Whirlwind—the biggest computer project of its time—becomes operational. Whirlwind is not only fast, but uses only 400 vacuum tubes (compared to the nearly 18,000 in ENIAC).

**1953**
After spending four years in development, Jay Forrester and a team at the Massachusetts Institute of Technology install magnetic core memory into the Whirlwind computer, giving it a twice-as-fast access time of six microseconds.

**1954**
Texas Instruments announces the start of commercial production of silicon transistors.

**1956**
The Nobel Prize in physics is awarded to John Bardeen, Walter Brattain, and William Shockley for their work on the transistor.

**1956**
The first transistorized computer is completed, the TX-O (Transistorized Experimental computer), at the Massachusetts Institute of Technology.

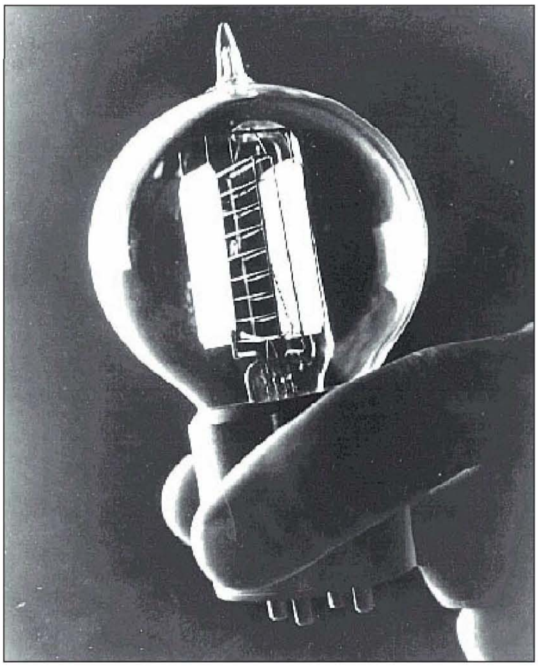
**1958**
At Texas Instruments, Jack Kilby comes up with the idea of creating a monolithic device, an integrated circuit (IC) that would contain resistors and capacitors on a single piece of silicon. Kilby builds the first integrated circuit which contains five components connected by wires on a sliver of germanium half an inch long and thinner than a toothpick.

**1959**
Robert Noyce of Fairchild Semiconductors seeks a patent for a new invention: an integrated circuit with components connected by aluminum lines on a silicon-oxide surface layer on a plane of silicon.

**1960**
IBM develops the first automatic mass-production facility for transistors, in New York.

Essentially, a computer is just a collection of on/off switches, which at first doesn't seem very useful. But imagine a large array of light bulbs—say, 10 rows that each have 100 light bulbs in them. Each bulb is connected to a light switch. If you turn on the right combination of switches, you can put your name in lights.

Computers are similar to that bank of lights, with one important difference: A computer can sense which light bulbs are on and use that information to turn on other switches. If the pattern of on switches spells "Tom," the computer could be programmed to associate the Tom pattern with instructions to turn on another group of switches to spell "boy." If the pattern spells "Mary," the computer could turn on a different group of switches to spell "girl."



The first vacuum tubes, such as this one made in 1915, were used to amplify radio signals. It wasn't until 1939 that tubes were used as switches in calculating machines.
***Courtesy of AT&T***

The two-pronged concept of On and Off maps perfectly with the binary number system, which uses only 0 and 1 to represent all numbers. By manipulating a roomful of vacuum tubes, early computer engineers could perform binary mathematical calculations. By assigning alphanumeric characters to certain numbers, they could manipulate text.

The problem with those first computers, however, was that the intense heat generated by the hundreds of vacuum tubes made them notoriously unreliable. The heat caused many components to deteriorate and consumed enormous amounts of power. But for vacuum tubes to work as switched, the tubes didn't really need to generate the immense flow of electrons that they created. A small flow would do quite nicely, but vacuum tubes were big. They worked on a human scale in which each part could be seen with the naked eye. They were simply too crude to produce more subtle flows of electrons. Transistors changed the way computers could be built.

A **transistor** is essentially a vacuum tube built, not to human size, but on a microscopic scale. Because it's small, a transistor requires less power to generate a flow of electrons. Because it uses less power, a transistor generates less heat, making computers more dependable. And the microscopic scale of transistors means that a computer that once

**1960**
The first integrated circuits reach the market, costing $120. NASA selects Noyce's invention for the on-board computers of the Gemini spacecraft.

**1961**
Fairchild Semiconductor releases the first commercial integrated circuit.

**1964**
The first integrated circuit sold commercially is used in a Zenith hearing aid.

**1964**
Intel founder Gordon Moore suggests that integrated circuits would double in complexity every year. This later becomes known as Moore's Law.

**1968**
Intel Corporation is founded in Santa Clara, CA, by Fairchild veterans Robert Noyce and Gordon Moore, employees #1 and #2. Andy Grove leaves Fairchild to become Intel's employee #4.

**1969**
Advanced Micro Devices Incorporated is founded.

**1969**
Intel's Marcian (Ted) Hoff designs an integrated circuit chip that can receive instructions and perform simple functions on data. Intel also announces a 1K RAM chip, a significantly larger capacity for memory chips.

**1969**
Intel sells its first commercial product, the 3101 Schottky bipolar 64-bit static random access memory (SRAM) chip. It is moderately successful.

took up an entire room now fits neatly on your lap. All microchips, whether they're microprocessors, a memory chip, or a special-purpose integrated circuit, are basically vast collections of transistors—switches—arranged in different patterns so that they accomplish different tasks. Doesn't sound like much but it's turning out to be nearly everything.

## KEY CONCEPTS

**adder, half-adder, full-adder** Differing combinations of transistors perform mathematical and logical operations on data being processed.

**address line** An electrical line, or circuit, associated with a specific location in RAM.

**arithmetic logic unit (ALU)** The central part of a microprocessor that manipulates the data received by the processor.

**ASCII** Acronym for American Standard Code for Information Interchange.

**binary** Consisting of only two integers, 0 and 1. Binary math is the basis for manipulating all data in computers.

**Boolean operations** Logical operations, based on whether a statement is true or false, that are the equivalent of mathematical operations with numbers.

**bunny suit** A total-body garment worn by personnel in a clean-room to reduce release of particles and contaminants into the air.

**burn-in** The process of exercising an integrated circuit at elevated voltage and temperature. This process accelerates failure normally seen as "infant mortality" in a chip. The resultant tested product is of high quality.

**cache** A block of high-speed memory where data is copied when it is retrieved from RAM. Then, if the data is needed again, it can be retrieved from the cache faster than from RAM. A *Level 1* cache is located on the CPU die. A *Level 2* is either a part of the processor die or packaging.

**capacitor** A component that stores an electrical charge.

**complex instruction set computing (CISC)** A processor architecture design in which large, complicated instructions are broken down into smaller tasks before the processor executes them. See *reduced instruction set computing*.

**data line** An electrical line, or circuit, that carries data; specifically in RAM chips, a circuit that determines whether a bit represents a 0 or a 1.

**drain** The part of a transistor where electrical current flows out when it is closed.

**gate** A microcircuit design in which transistors are arranged so the value of a bit of data can be changed.

**logic** A collection of circuit elements that perform a function, especially a set of elements that use digital logic and perform Boolean logic functions.

**1971**
Intel introduces its 4-bit bus, 108-KHz 4004 chip—the first microprocessor. Initial price is $200. Speed is 60,000 operations per second. It uses 2,300 transistors connected by circuits 10 microns wide. It can address 640 bytes of memory. The dimensions for the chip are 3×4 mm.

**1972**
Intel introduces the 8008, the first 8-bit microprocessor. Don Lancaster, a dedicated computer hobbyist, used the 8008 to create a predecessor to the first personal computer, a device Radio Electronics dubbed a "TV typewriter." It was used as a dumb terminal.

**1974**
The Intel 8080 microprocessor becomes the brains of the first personal computer: the Altair. Computer hobbyists could purchase a kit for the Altair for $367.

**1975**
The January edition of *Popular Electronics* features on its cover the Altair 8800 computer kit, based on Intel's 8080 microprocessor. Within months, it sells tens of thousands, creating the first PC back orders in history. Bill Gates and Paul Allen licensed BASIC as the software language for the Altair.

**1979**
A pivotal sale to IBM's new personal computer division makes the Intel 8088 processor the brains of IBM's new hit product: the IBM PC.

**logic design** Techniques used to connect logic building blocks or primitives (that is, AND gates, OR gates, and so on) to perform a logical operation.

**megahertz (MHz)** A measurement, in millions, of the number of times something oscillates or vibrates. Processor speeds are normally measured in gigahertz (GHz).

**microchip** A sheet of silicon dioxide on microscopic electrical circuits that have been etched using a system of light, light-sensitive films, and acid baths.

**micrometer** A metric unit of linear measure that equals 1/1,000,000 meter, or 10,000 angstroms. The width of microprocessor circuits are measured in micrometers. The diameter of a human hair is approximately 75 micrometers. Also called "micron."

**microprocessor, processor** The "brains" of a computer. A component that contains circuitry that can manipulate data in the form of binary bits. A microprocessor is contained on a single microchip.

**pin** In plastic and metal wafer carriers, a protrusion of the wafer that fits into a matching hole in the wafer carrier for alignment when wafers are transferred.

**pin grid array (PGA)** A connection arrangement for microchips that features plug-in electrical terminal pins arranged in a matrix format, or an array.

**pipelining** A computer architecture designed so that all parts of a circuit are always working, and that no part of the circuit is stalled—waiting for data from another part.

**reduced instruction set computing (RISC)** A processor design in which only small, quickly executing instructions are used. Contrast to *complex instruction set computing.*

**register** A set of transistors in a processor where data is stored temporarily while the processor makes calculations involving that data—a sort of electronic scratch pad.

**SIMD (Single Instruction Multiple Data)** A processor architecture that allows the same operation to be performed on multiple pieces of data simultaneously.

**semiconductor** A material (such as silicon) that can be altered to either conduct electrical current or block its passage. Microchips are typically fabricated on semiconductor materials such as silicon, germanium, or gallium arsenide.

**silicon** A brownish crystalline semimetal used to make the majority of semiconductor wafers.

**source** The part of a transistor from which electrical current flows when the transistor is closed.

**transistor** A microscopic switch that controls the flow of electricity through it, depending on whether a different electrical charge has opened or closed the switch.

**wafer** In semiconductor technology, a very thin piece of silicon that functions as the base material for building microchips. Also called a "slice."

**1982**
The Intel 80286 is the first Intel processor that can run all the software written for its predecessor. This software compatibility remains a hallmark of Intel's family of microprocessors. Within six years of its release, an estimated 15 million 286-based personal computers are installed around the world.

**1985**
Motorola announces the 68040, a 32-bit 25MHz microprocessor.

**1985**
The Intel 80386 microprocessor features 275,000 transistors—more than 100 times as many as the original 4004. It handles data 32 bits at a time and multitasks, meaning it can run multiple programs at the same time.

**1989**
The Intel 80486 DX makes point-and-click computing practical. The 486 is the first processor to offer a built-in math coprocessor, which speeds up computing because it offers complex math functions from the central processor.

**1991**
Advanced Micro Devices introduces the AM386 microprocessor family in direct competition with Intel's x86 processor line.

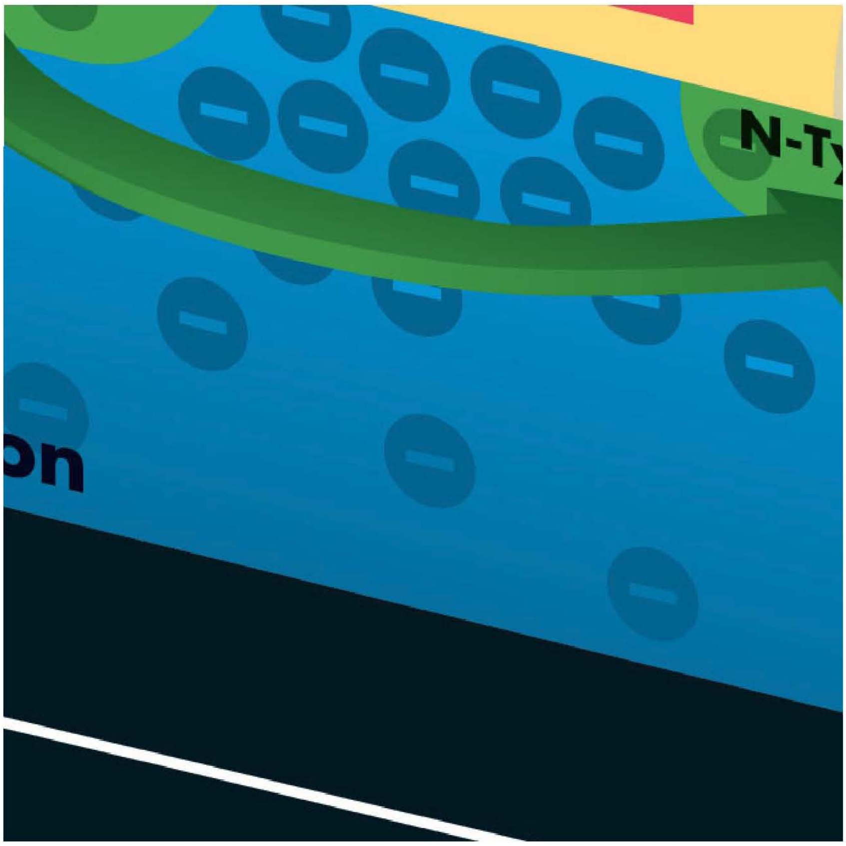
**1993**
Intel's new Pentium processor allows computers to more easily incorporate real-world data such as speech, sound, handwriting, and photographic images.

**2003**
AMD introduces the Athlon 64, the first 64-bit processor targeted for use in home computers.

**2005**
Both Intel and AMD release their first multicore processors.

# CHAPTER 5

# How Transistors Manipulate Data



**THE** transistor is the basic building block from which all microchips are built. The transistor can only create binary information: a 1 if current passes through, or a 0 if current doesn't pass through. From these 1s and 0s, called **bits**, a computer can create any number as long as it has enough transistors grouped together to hold all the 1s and 0s.

Binary notation starts off simply enough:

| Decimal Number | Binary Number | Decimal Number | Binary Number |
|---|---|---|---|
| 0 | 0 | 6 | 110 |
| 1 | 1 | 7 | 111 |
| 2 | 10 | 8 | 1000 |
| 3 | 11 | 9 | 1001 |
| 4 | 100 | 10 | 1010 |
| 5 | 101 | | |

Personal computers, such as the original IBM PC and AT systems based on the Intel 8088 and 80286 microprocessors, are 16-bit PCs. That means they can work directly with binary numbers of up to 16 places, or bits. That translates to the decimal number 65,535. If an operation requires numbers larger than that, the PC must first break those numbers into smaller components, perform the operation on each component, and then recombine the results into a single answer. More powerful PCs, such as those based on the Intel 80386, 80486, and Pentium, are 32-bit computers, which means they can manipulate binary numbers up to 32 bits wide—the equivalent in decimal notation of 4,294,967,295. The capability to work with 32 bits at a time helps make these PCs much faster and capable of directly using more memory.

Transistors are not used simply to record and manipulate numbers. The bits can just as easily stand for true (1) or not true (0), which allows computers to deal with Boolean logic. ("Select this AND this but NOT this.") Combinations of transistors in various configurations are called **logic gates**, which are combined into arrays called **half adders**, which in turn are combined into **full adders**. More than 260 transistors are needed to create a full adder that can handle mathematical operations for 16-bit numbers.

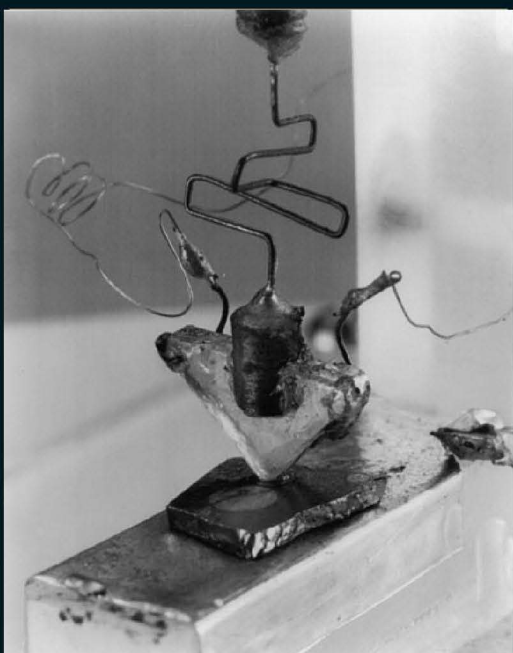
In addition, transistors make it possible for a small amount of electrical current to control a second, much stronger current—just as the small amount of energy needed to throw a wall switch can control the more powerful energy surging through the wires to give life to a spotlight.

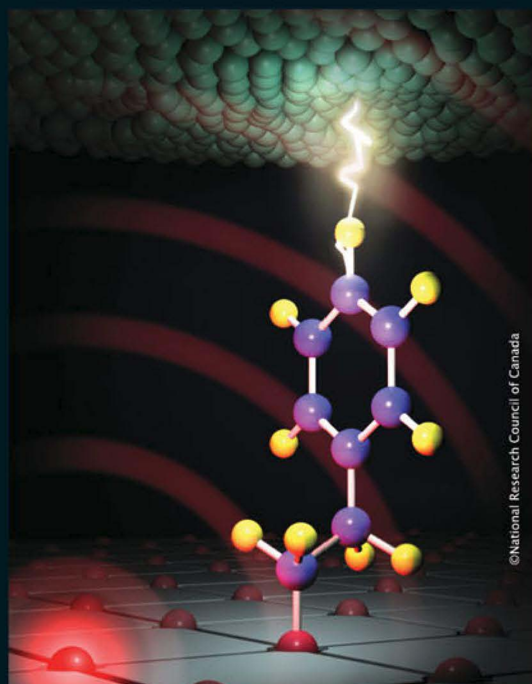
# How a Little Transistor Does Big Jobs

**1** A small, positive electrical charge is sent down one aluminum lead that runs into the transistor. The positive charge spreads to a layer of electrically conductive polysilicon buried in the middle of nonconductive silicon dioxide. Silicon dioxide is the main component of sand and the material that gave Silicon Valley its name.

**2** The positive charge attracts negatively charged electrons from the base made of P-type (positive) silicon that separates two layers of N-type (negative) silicon.

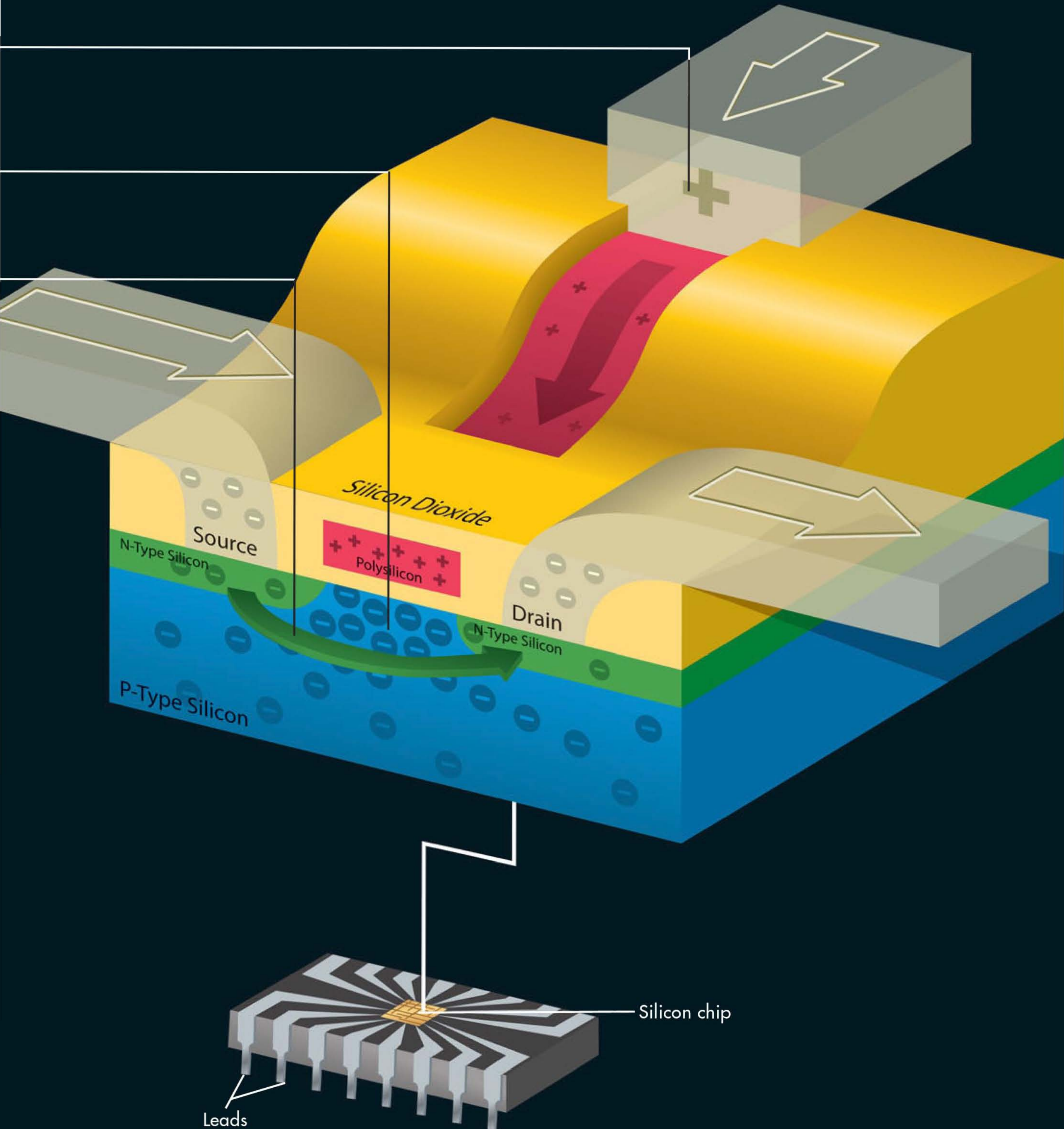
**3** The rush of electrons out of the P-type silicon creates an electronic vacuum that is filled by electrons rushing from another conductive lead called the *source*. In addition to filling the vacuum in the P-type silicon, the electrons from the source also flow to a similar conductive lead called the *drain*. The rush of electrons completes the circuit, turning on the transistor so that it represents 1 bit. If a negative charge is applied to the polysilicon, electrons from the source are repelled and the transistor is turned off.






### Creating a Chip from Transistors

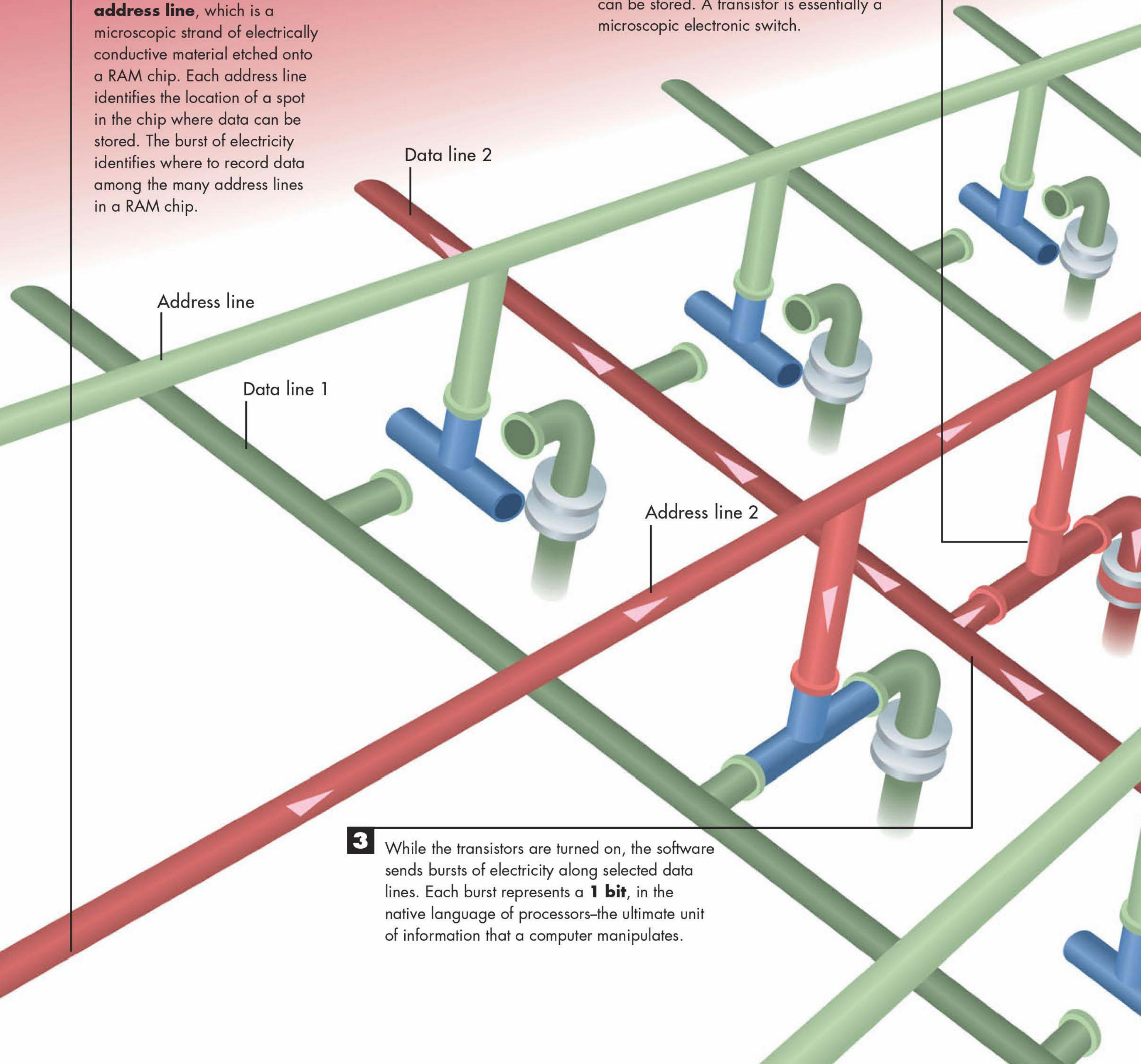
Thousands of transistors are connected on a single slice of silicon. The slice is embedded in a piece of plastic or ceramic material and the ends of the circuitry are attached to metal leads that expand to connect the chip to other parts of a computer circuit board. The leads carry signals into the chip and send signals from the chip to other computer components.


Silicon Dioxide
Source
N-Type Silicon
Polysilicon
Drain
N-Type Silicon
P-Type Silicon
Silicon chip
Leads

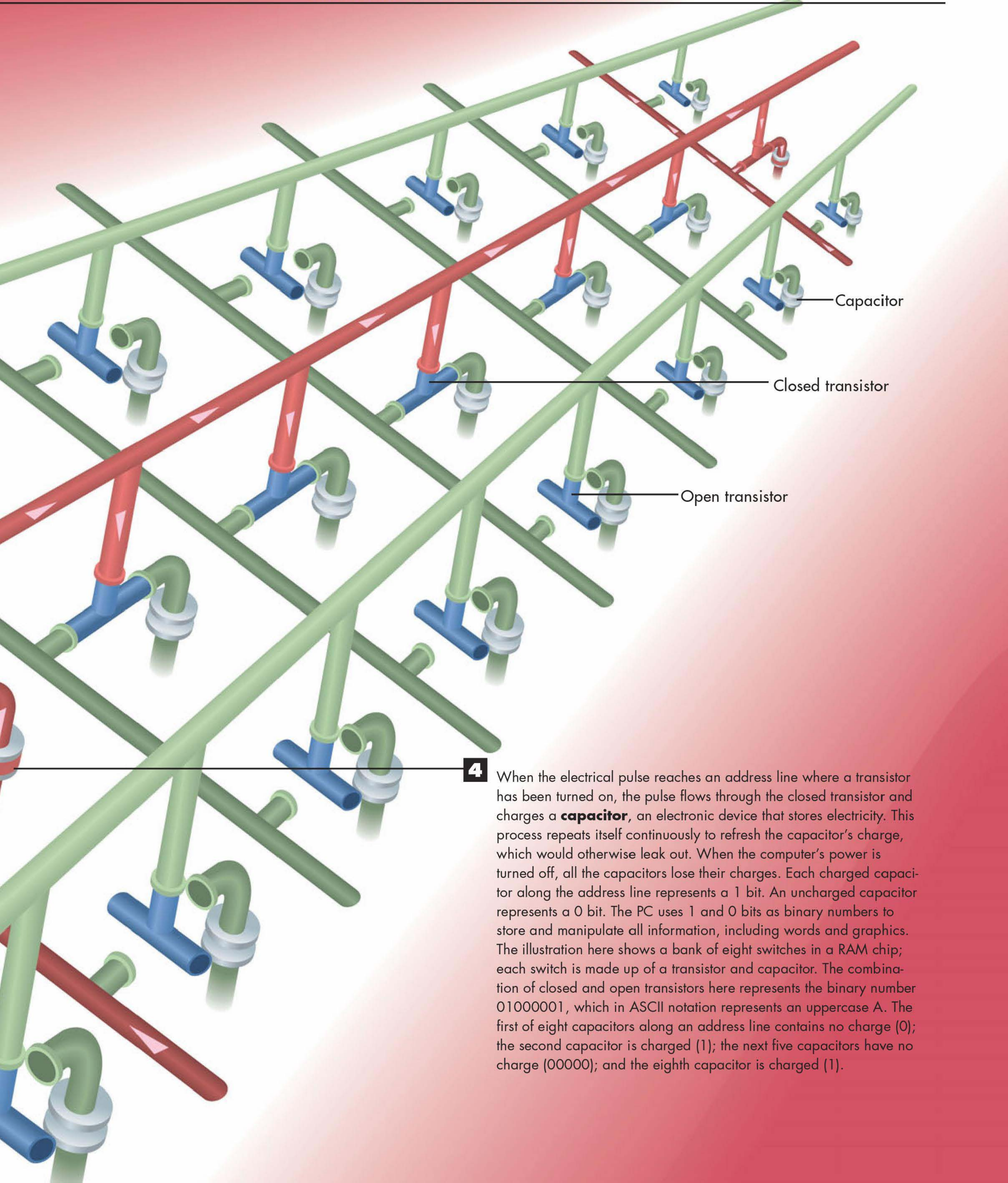
# Writing Data to RAM

**1** Software, in combination with the operating system, sends a burst of electricity along an **address line**, which is a microscopic strand of electrically conductive material etched onto a RAM chip. Each address line identifies the location of a spot in the chip where data can be stored. The burst of electricity identifies where to record data among the many address lines in a RAM chip.

**2** The electrical pulse turns on (closes) a transistor that's connected to a **data line** at each memory location in a RAM chip where data can be stored. A transistor is essentially a microscopic electronic switch.




**3** While the transistors are turned on, the software sends bursts of electricity along selected data lines. Each burst represents a **1 bit**, in the native language of processors–the ultimate unit of information that a computer manipulates.

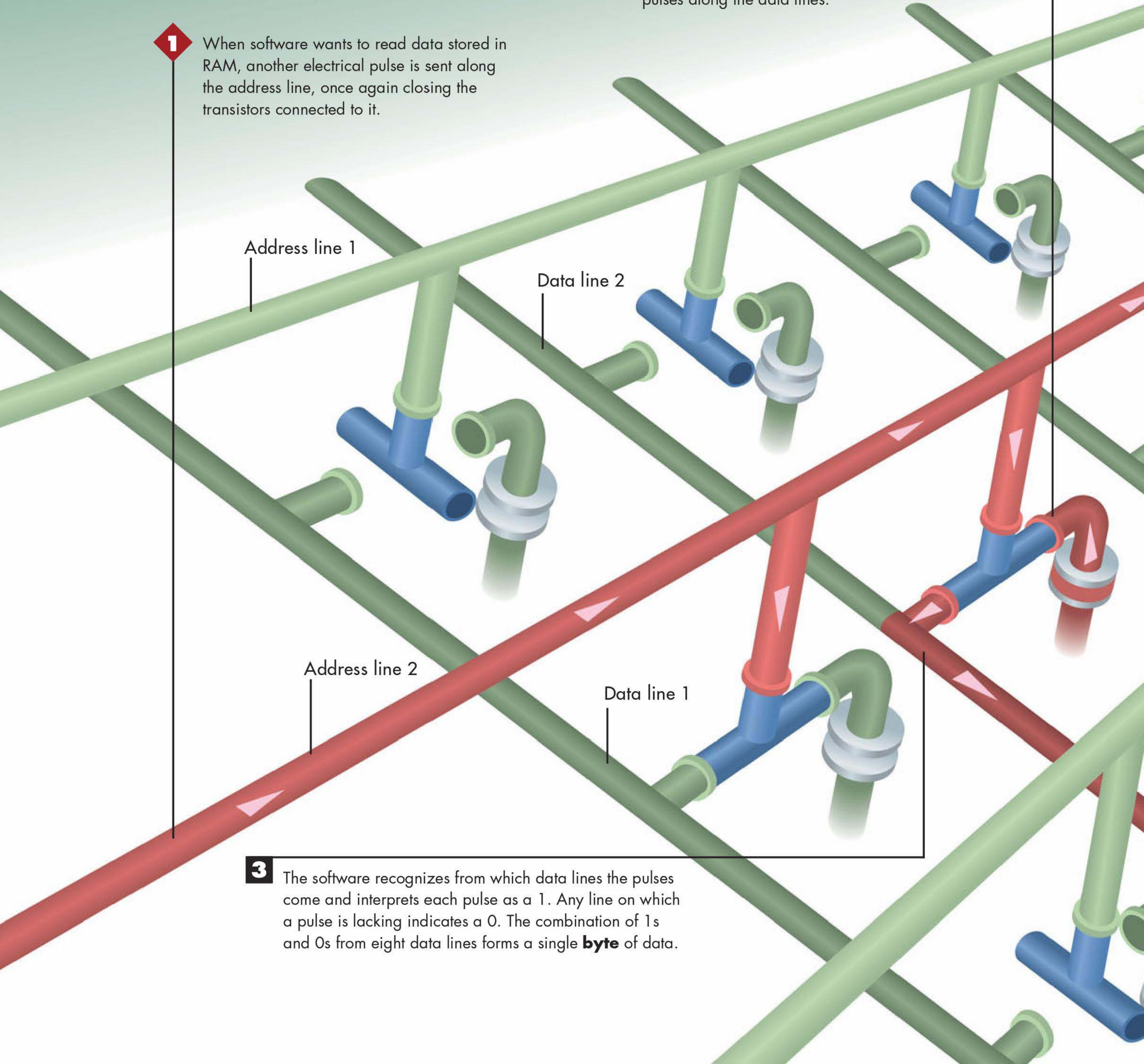



**4** When the electrical pulse reaches an address line where a transistor has been turned on, the pulse flows through the closed transistor and charges a **capacitor**, an electronic device that stores electricity. This process repeats itself continuously to refresh the capacitor's charge, which would otherwise leak out. When the computer's power is turned off, all the capacitors lose their charges. Each charged capacitor along the address line represents a 1 bit. An uncharged capacitor represents a 0 bit. The PC uses 1 and 0 bits as binary numbers to store and manipulate all information, including words and graphics. The illustration here shows a bank of eight switches in a RAM chip; each switch is made up of a transistor and capacitor. The combination of closed and open transistors here represents the binary number 01000001, which in ASCII notation represents an uppercase A. The first of eight capacitors along an address line contains no charge (0); the second capacitor is charged (1); the next five capacitors have no charge (00000); and the eighth capacitor is charged (1).

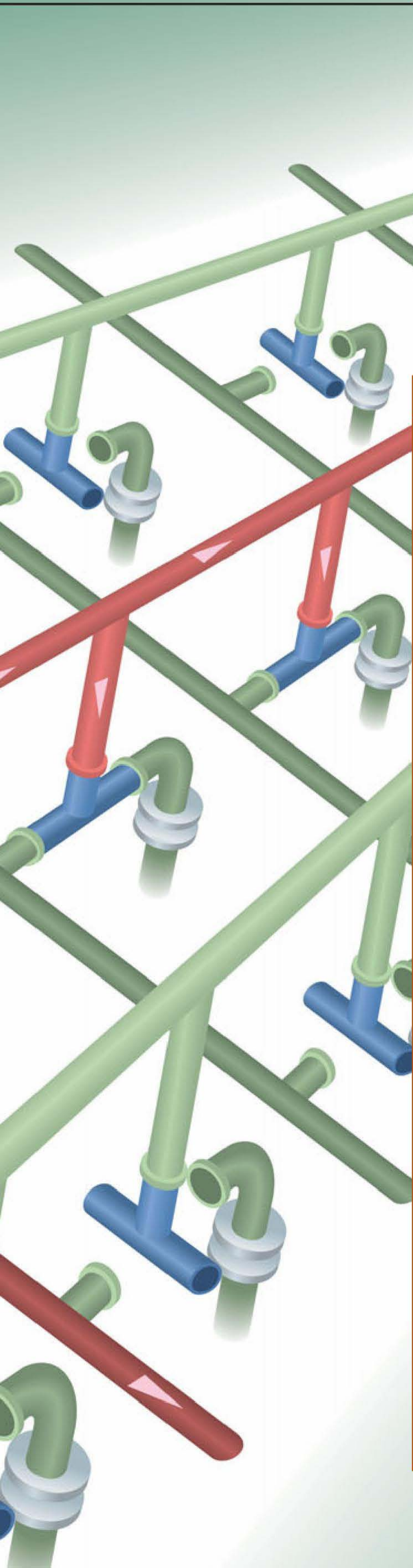
# Reading Data from RAM

**2** Everywhere along the address line that there is a capacitor holding a charge, the capacitor will discharge through the circuit created by the closed transistors, sending electrical pulses along the data lines.

**1** When software wants to read data stored in RAM, another electrical pulse is sent along the address line, once again closing the transistors connected to it.




**3** The software recognizes from which data lines the pulses come and interprets each pulse as a 1. Any line on which a pulse is lacking indicates a 0. The combination of 1s and 0s from eight data lines forms a single **byte** of data.

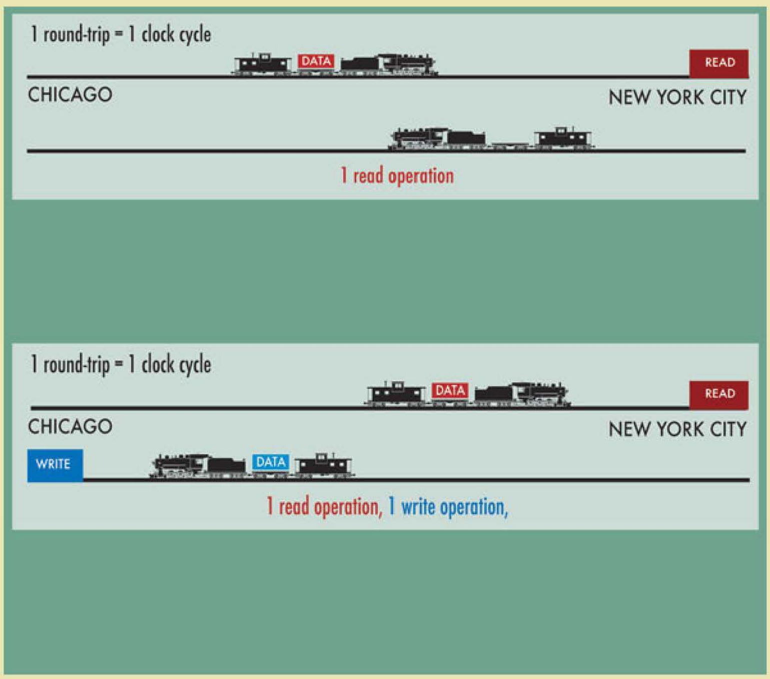


## How DDR2 RAM Doubles Times

The fastest processors are limited by how fast memory feeds them data. Traditionally, the way to pump out more data was to increase the clock speed. With each cycle, or tick, of the clock regulating operations in the processor and movement of memory data, SDRAM memory—the kind illustrated here—could store a value or move a value out and onto the data bus headed to the processor. But the speeds of processors outstripped that of RAM. Memory caught up to processors two ways.

One is **double data rate (DDR)**. Previously, a bit was written or read on each cycle of the clock. It's as if someone loaded cargo (writing data) onto a train traveling from Chicago to New York, unloaded that cargo (reading), and then had to send the empty train back to Chicago again, despite having fresh cargo in New York that could hitch along for the return trip. With DDR, a handler could unload that same cargo when the train arrives in New York and then load it back up with new cargo again before the train makes its journey back to Chicago. This way, the train is handling twice as much traffic (data) in the same amount of time. Substitute *memory controller* for the persons loading and unloading cargo and *clock cycle* for each round-trip of the train, and you have DDR.

The other method is **dual channel architecture**—the 2 in DDR2. With double data rates alone, there are times when no data is ready to be stored or read from memory locations. It's as if the train reached one end of its line and the handler there hadn't found any cargo to put on the train. Dual channel adds another pipeline to supply memory to help ensure there is data for the train.



# How Memory Cards and Flash Drives Work

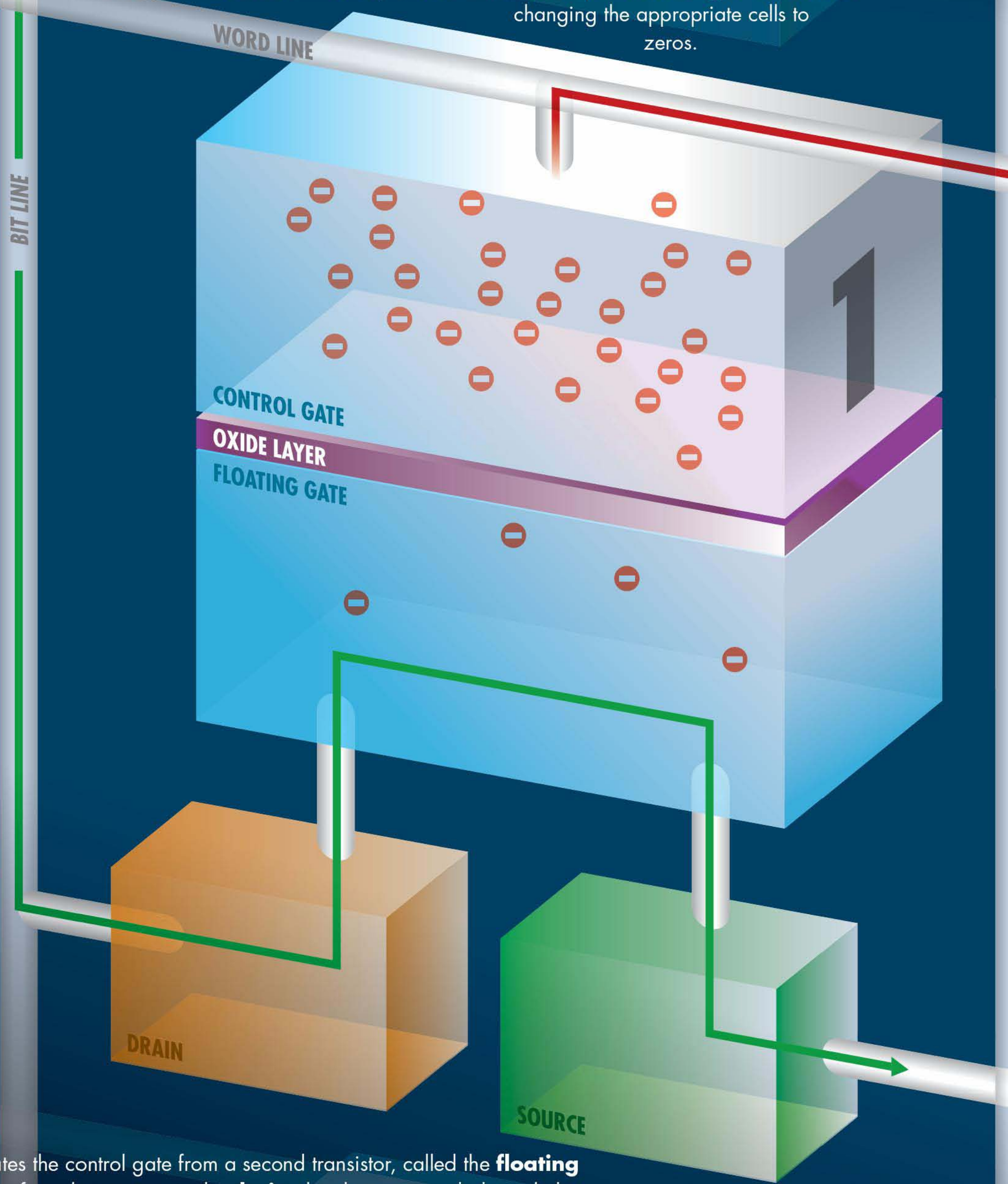
**1** You know what happens when you turn off your PC—anything in RAM disappears. The next time you turn it on, the PC's memory is a blank slate, ready take on the form of the programs you run. **Flash memory** is different. When you turn off the computer, camera, phone, or MP3 player using flash memory, the documents, photos, numbers, and songs are still there when you turn it back on.

**2** Flash memory is laid out along a grid of printed circuits running at right angles to each other. In one direction, the circuit traces are **word addresses**; circuits at a right angle to them represent the **bit addresses**. The two addresses combine to create a unique number address called a **cell**.

**3** The cell contains two transistors that together determine if an intersection represents a 0 or a 1. One transistor—the **control gate**—is linked to one of the passing circuits called the **word line**, which determines the word address.

**4** A thin layer of **metal oxide** separates the control gate from a second transistor, called the **floating gate**. When an electrical charge runs from the **source** to the **drain**, the charge extends through the floating gate, on through the metal oxide, and through the control gate to the word line.

**5** A **bit sensor** on the word line compares the strength of the charge in the control gate to the strength of the charge on the floating gate. If the control voltage is at least half of the floating gate charge, the gate is said to be **open**, and the cell represents a 1. Flash memory is sold with all cells open. Recording to it consists of changing the appropriate cells to zeros.

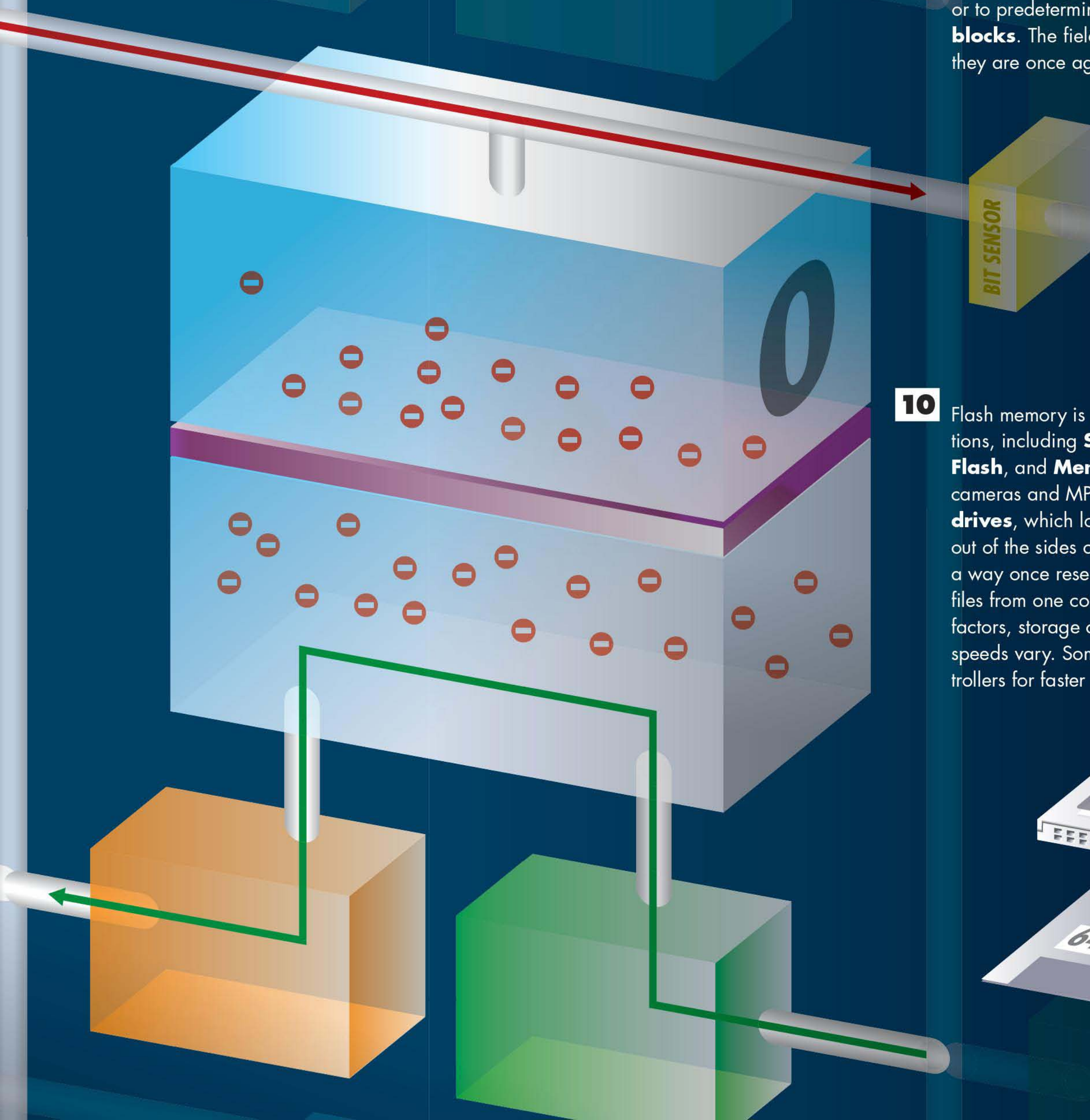


**6** Flash memory uses **Fowler-Nordheim tunneling** to change the value of the cell to a zero. In tunneling, a current from the bit line travels through the floating gate transistor, exiting through the source to a **ground**.

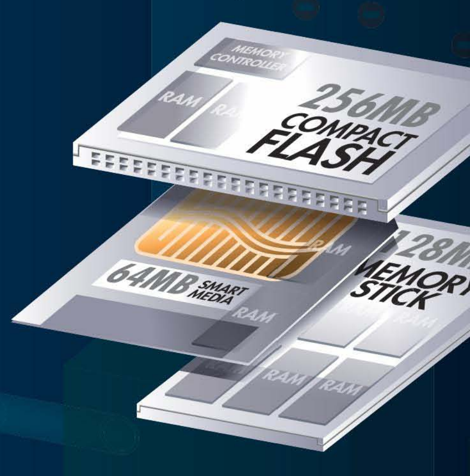
**7** Energy from the current causes electrons to boil off the floating gate and through the metal oxide, where the electrons lose too much energy to make it back through the oxide. They are trapped in the control gate, even when the current is turned off.

**8** The electrons have become a wall that repels any charge coming from the floating gate. The bit sensor detects the difference in charges on the two transistors, and because the charge on the control gate is below 50 percent of the floating gate charge, it is considered to stand for a zero.

**9** When it comes time to reuse the flash memory, a current is sent through **in-circuit wiring** to apply a strong electrical field to the entire chip, or to predetermined sections of the chip called **blocks**. The field energizes the electrons so they are once again evenly dispersed.




**10** Flash memory is sold in a variety of configurations, including **SmartMedia**, **Compact Flash**, and **Memory Sticks**, used mainly in cameras and MP3 players. There's also **flash drives**, which look like gum packages sticking out of the sides of computers, and are handy in a way once reserved for floppies for moving files from one computer to another. The form factors, storage capacities, and read/write speeds vary. Some include their own controllers for faster reads and writes.



CHAPTER

6

# How a Microprocessor Works

**THE** easiest way to visualize how computers work is to think of them as enormous collections of switches, which is really what they are—switches in the form of microscopic transistors etched into a slice of silicon. But for the moment, think of a computer as a giant billboard made up of columns and rows of lights—thousands of them. Then imagine a control room behind that billboard in which there is a switch for every one of the light bulbs on the sign. By turning on the correct switches, you can spell your name or draw a picture.

But suppose there are "master switches" that control dozens of other switches. Instead of having to flip each switch individually for every light bulb that goes into spelling your name, you can throw one switch that lights up a combination of lights to create a B, then another master switch that turns on all the lights for an O, and another switch to light up another B.

Now you're very close to understanding how a computer works. In fact, substitute a computer display for the billboard, and substitute RAM—which is a collection of transistorized switches—for the control room, and a keyboard for the master switches, and you have a computer performing one of its most basic functions: displaying what you type onscreen.

A computer must do a lot more than display words to be helpful. But the off and on positions of the same switches used to control a display can also add numbers by representing the 0 and 1 in the binary number system. If you can add numbers, you can perform any kind of math because multiplication is simply repeated addition, subtraction is adding a negative number, and division is repeated subtraction. To a computer, everything—math, words, numbers, and software instructions—is numbers. This fact lets all those switches (transistors) do all types of data manipulation.

Actually, the first computers were more like our original billboard in how they were used. They didn't have keyboards or displays. The first computer users actually did throw a series of switches in a specific order to represent both data and the instructions for handling that data. Instead of transistors, the early computers used vacuum tubes, which were bulky and generated an enormous amount of heat. To get the computer's answer, the people using it had to decipher what looked like a random display of lights. Even with the most underpowered PC you can buy today, you still have it a lot better than the earliest computer pioneers.

## The Brains

The microprocessor that makes up your personal computer's **central processing unit**, or CPU, is the ultimate computer brain, messenger, ringmaster, and boss. All the other components—RAM, disk drives, the display—exist only to bridge the gap between you and the processor. They take your data and turn it over to the processor to manipulate; then they display the results. The CPU isn't the

only microprocessor in PCs. Coprocessors on graphics, 3D accelerators, and sound cards juggle display and sound data to relieve the CPU of part of its burden. And special processors, such as the one inside your keyboard that handles the signals generated whenever you press a key, perform specialized tasks designed to get data into or out of the CPU.

The first processor in an IBM PC was Intel's 8088 (the CPU itself was a follow-up to Intel's 8086). The generations of Intel processors that followed it were in the 80x86 family—80286, 80386, and 80486. All were more elaborate versions of the original 8088, but improved on its performance by one of two ways: operating faster or handling more data simultaneously. The 8088, for example, operated at 4.7MHz, or 4.7 million frequency waves a second; some 80486 chips go as fast as 133MHz. The 8088 could handle 8 bits of data at a time, and the 80486 handles 32 bits internally.

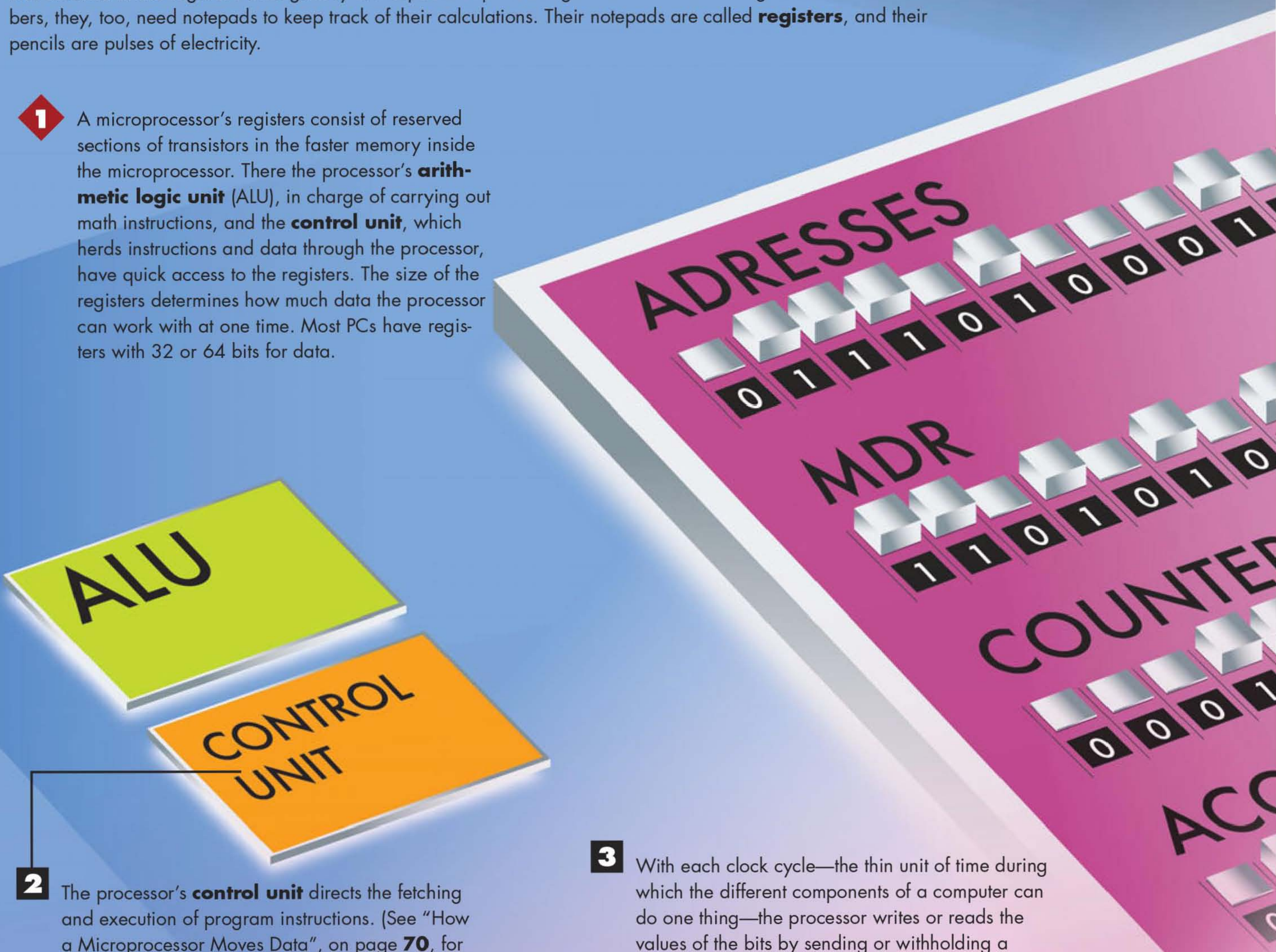
Intel and Advanced Micro Devices (AMD) are the only companies that make processors for Windows-based personal computers. The current standard for Intel processors is the Core 2 chip, the most recent being the Core 2 Quad. The combined chips cover less than a couple of square inches but hold more than 582 million **transistors**. All the operations of the Core 2 are performed by signals turning on or off different combinations of those switches. In computers, transistors are used to represent zeros and ones, the two numbers that make up the binary number system. These zeros and ones are commonly known as **bits**. Various groupings of these transistors make up the subcomponents within the Core 2, as well as those in coprocessors, memory chips, and other forms of digital silicon.

There are Core 2 processors designed to fill every market niche, from the bargain basement to the network server room. At the lowest end are Celeron processors with limited internal cache. They provide the function of the Pentium architecture with less speed. At the high end are Extreme Editions, which include large caches, and move data more quickly between it and the motherboard's chipset.

# How the Processor Uses Registers

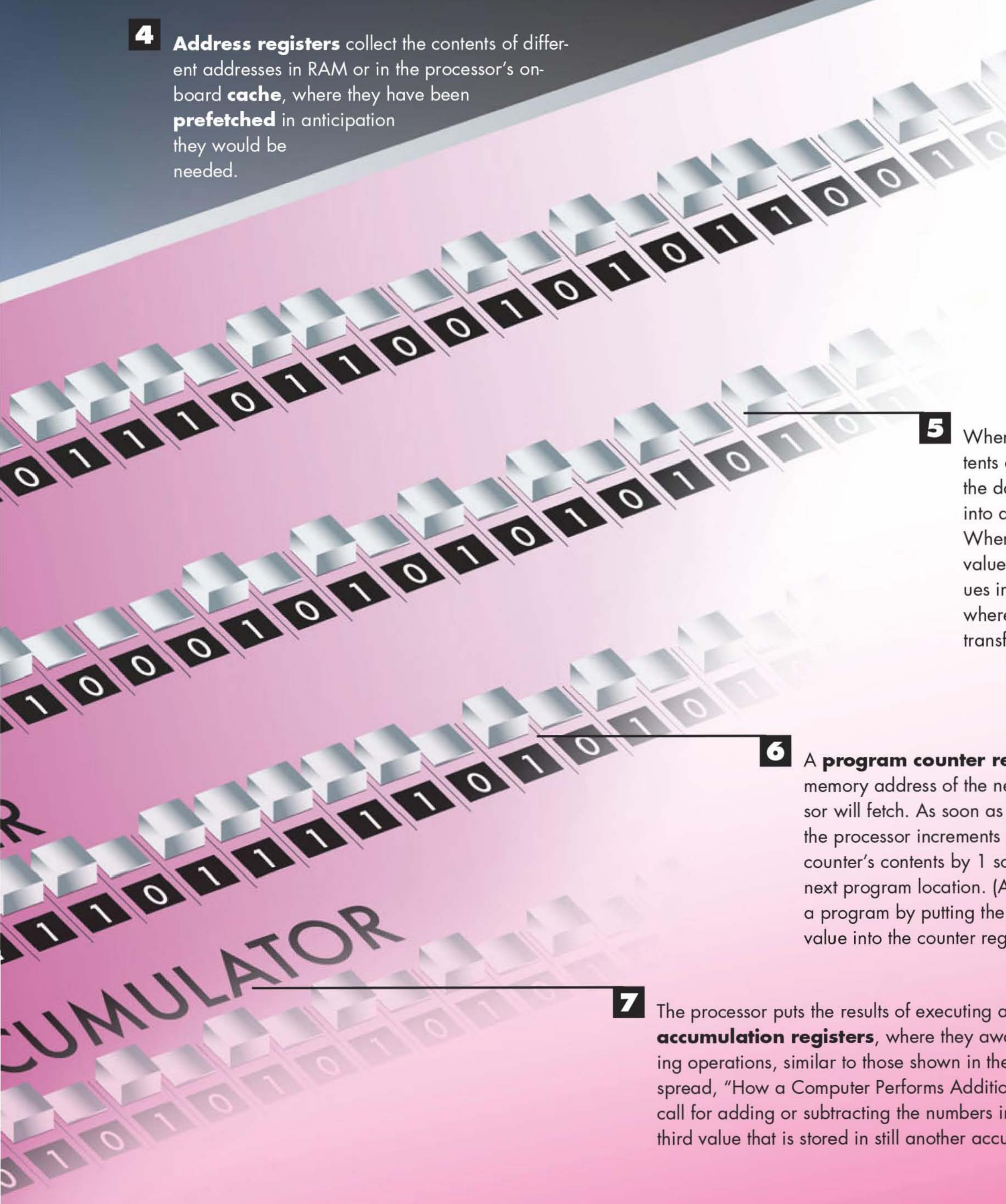
Few of us can do complex math in our heads. Even for something as simple as adding several rows of numbers, we need a pencil and paper to keep track of our operations on individual numbers. Microprocessors are not all that different in this regard. Although they are capable of performing intricate math involving thousands of numbers, they, too, need notepads to keep track of their calculations. Their notepads are called **registers**, and their pencils are pulses of electricity.

1. A microprocessor's registers consist of reserved sections of transistors in the faster memory inside the microprocessor. There the processor's **arithmetic logic unit** (ALU), in charge of carrying out math instructions, and the **control unit**, which herds instructions and data through the processor, have quick access to the registers. The size of the registers determines how much data the processor can work with at one time. Most PCs have registers with 32 or 64 bits for data.




2. The processor's **control unit** directs the fetching and execution of program instructions. (See "How a Microprocessor Moves Data", on page **70**, for more information.) It uses an electrical signal to fetch each instruction, decodes it, and sends another control signal to the arithmetic logic unit telling the ALU what operation to carry out.

3. With each clock cycle—the thin unit of time during which the different components of a computer can do one thing—the processor writes or reads the values of the bits by sending or withholding a pulse of electricity to each bit. Each chunk of binary numbers is only that. They have no labels to identify them as instructions, data, values going into a computation, or the product of executing instructions. What the values represent depends on in which registers the control unit stores them.




**4** **Address registers** collect the contents of different addresses in RAM or in the processor's onboard **cache**, where they have been **prefetched** in anticipation they would be needed.

**5** When the processor reads the contents of a location in memory, it tells the data bus to place those values into a **memory data register**. When the processor wants to write values to memory, it places the values in the memory data register, where the bus retrieves them to transfer to RAM.

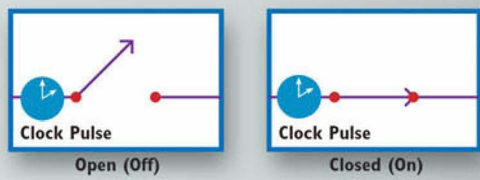
**6** A **program counter register** holds the memory address of the next value the processor will fetch. As soon as a value is retrieved, the processor increments the program counter's contents by 1 so it points to the next program location. (A computer launches a program by putting the program's first value into the counter register.)

**7** The processor puts the results of executing an operation into several **accumulation registers**, where they await the results of other executing operations, similar to those shown in the illustration on the next spread, "How a Computer Performs Addition." Some of the instructions call for adding or subtracting the numbers in two accumulators to yield a third value that is stored in still another accumulator.

# How a Computer Performs Addition

**1** All information—words and graphics as well as numbers—is stored in and manipulated by a PC in the form of binary numbers. In the binary numerical system, there are only two digits—0 and 1. All numbers, words, and graphics are formed from different combinations of those digits.

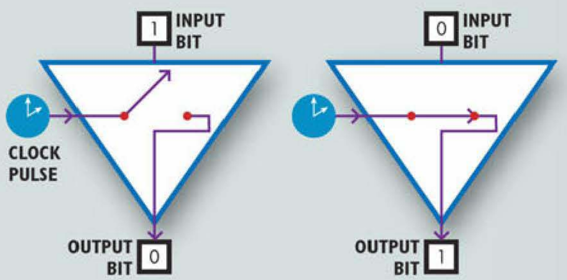
| Decimal | Binary |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |




**2** Transistor switches are used to manipulate binary numbers because there are two possible states of a switch, open (off) or closed (on), which nicely matches the two binary digits. An open transistor, through which no current is flowing, represents a 0. A closed transistor, which allows a pulse of electricity regulated by the PC's clock to pass through, represents a 1. (The computer's clock regulates how fast the computer works. The faster a clock ticks, causing pulses of electricity, the faster the computer works. Clock speeds are measured in *megahertz*, or millions of ticks per second.) Current passing through one transistor can be used to control another transistor, in effect turning the switch on and off to change what the second transistor represents. Such an arrangement is called a *gate* because, like a fence gate, the transistor can be open or closed, allowing or stopping current flowing through it.

**3** The simplest operation that can be performed with a transistor is called a NOT logic gate, made up of only a single transistor. This NOT gate is designed to take one *input* from the clock and one from another transistor. The NOT gate produces a single *output*—one that's always the opposite of the input from the transistor. When current from another transistor representing a 1 is sent to a NOT gate, the gate's own transistor switches open so that a pulse, or current, from the clock can't flow through it, which makes the NOT gate's output 0. A 0 input closes the NOT gate's transistor so that the clock pulse passes through it to produce an output of 1.

**NOT Gate**

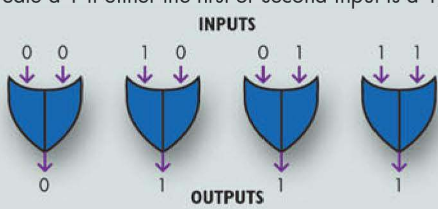



**NOT Gate Operations**

| INPUT FROM CLOCK | INPUT FROM OTHER TRANSISTOR | OUTPUT |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |

**4** NOT gates strung together in different combinations create other logic gates, all of which have a line to receive pulses from the clock and two other input lines for pulses from other logic gates. The OR gates create a 1 if either the first or second input is a 1, and put out a 0 only if both inputs are 0.

**OR Gate**

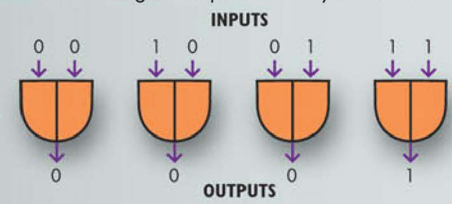



**OR Gate Operations**

| 1ST INPUT | 2ND INPUT | OUTPUT |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

**5** An AND gate outputs a 1 only if *both* the first and the second inputs are 1s.

**AND Gate**

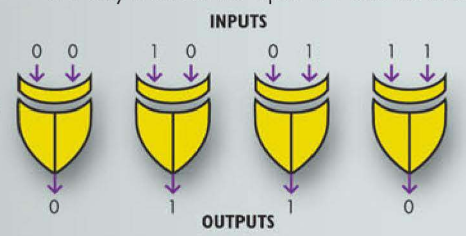



**AND Gate Operations**

| 1ST INPUT | 2ND INPUT | OUTPUT |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

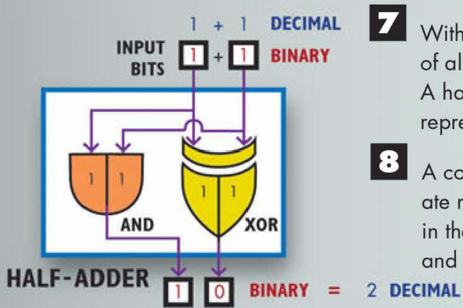
**6** An XOR gate puts out a 0 if *both* the inputs are 0 or if *both* are 1. It generates a 1 only if *one* of the inputs is 1 and the *other* is 0.

**XOR Gate**




**XOR Gate Operations**

| 1ST INPUT | 2ND INPUT | OUTPUT |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |




**7** With different combinations of logic gates, a computer performs the math that is the foundation of all its operations. This is accomplished with gate designs called *half-adders* and *full-adders*. A half-adder consists of an XOR gate and an AND gate, both of which receive the same input representing a one-digit binary number. A full-adder consists of half-adders and other switches.

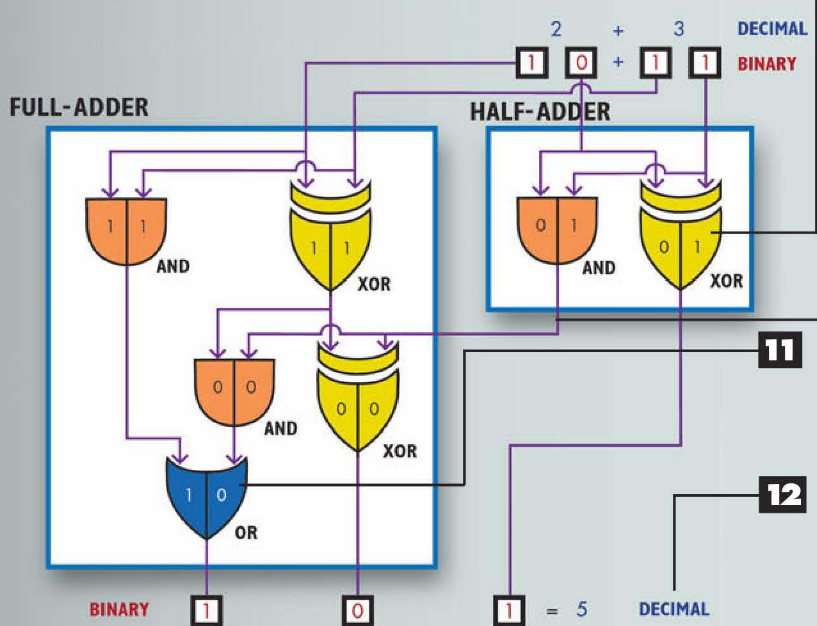
**8** A combination of a half-adder and a full-adder handles larger binary numbers and can generate results that involve carrying over numbers. To add the decimal numbers 2 and 3 (10 and 11 in the binary system), first the half-adder processes the digits on the right side through both XOR and AND gates.

**9** The result of the XOR operation (1) becomes the rightmost digit of the result.

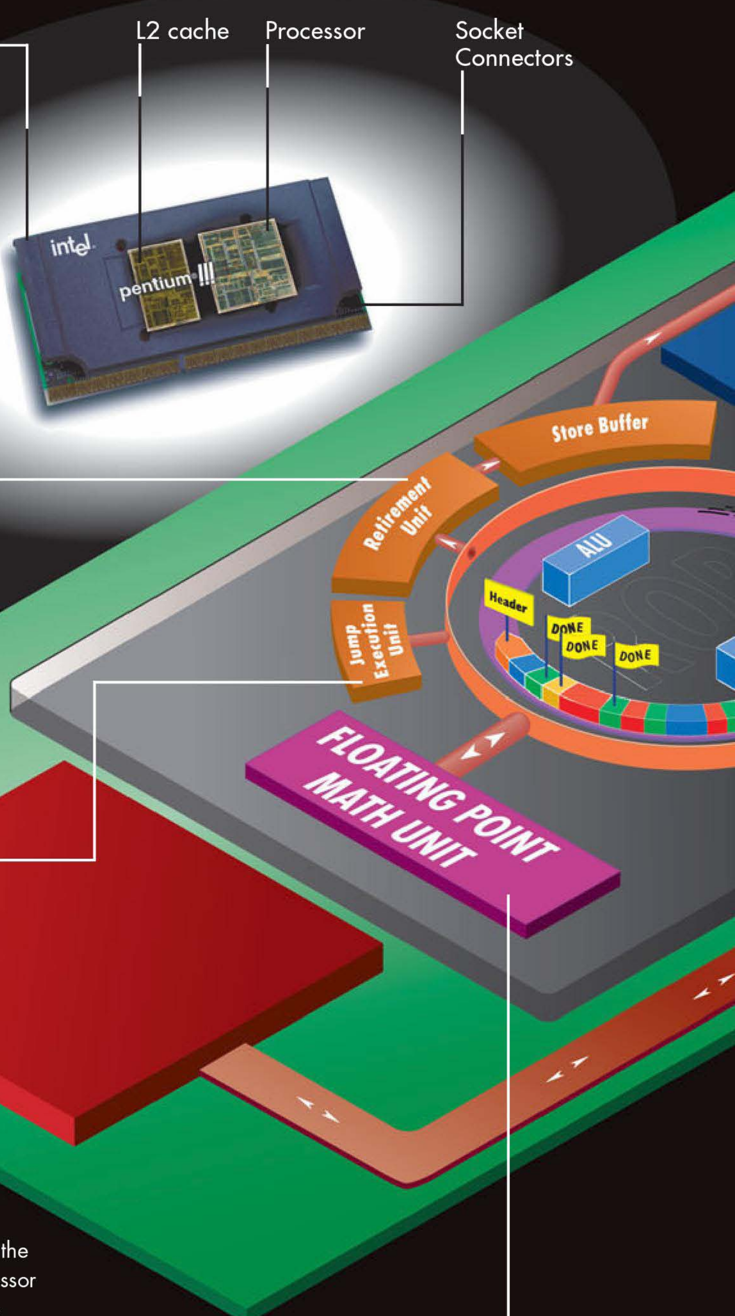
**10** The result of the half-adder's AND operation (0) is sent to XOR and AND gates in the full-adder. The full-adder also processes the left-hand digits from 11 and 10, sending the results of both of the operations to another XOR gate and another AND gate.

**11** The results from XORing and ANDing the left-hand digits are processed with the results from the half-adder. One of the new results is passed through an OR gate.

**12** The result of all the calculations is 101 in binary, which is 5 in decimal. For larger numbers, more full-adders are used—one for each digit in the binary numbers. An 80386 or later processor, including today's Pentium class processors, uses 32 full-adders.



# How a Microprocessor Moves Data




**1** Today's microprocessors have as many as 582 million transistors. Taking a walk through one of them could get a person hopelessly lost. And so let's return to a simpler time, when CPUs had a mere 5 million transistors and transacted business at a leisurely 133 MHz. This will give you an idea of the basic functions performed by all microprocessors. They may have two or four execution cores and multiple caches—and we'll look at those on pages **72–73**—but they all have at least one execution core and one cache, and they all face the same problem of how to move data quickly and with nary a hitch.

**12** Meanwhile, the retirement unit is also inspecting the circular buffer. It first checks to see whether the µop at the head of the buffer has been executed. If it hasn't, the retirement unit keeps checking it until it has been processed. Then, the retirement unit checks the second and third µops. If they're already executed, the unit sends all three results—its maximum—to the store buffer. There, the prediction unit checks them out one last time before they're sent to their proper place in system RAM.

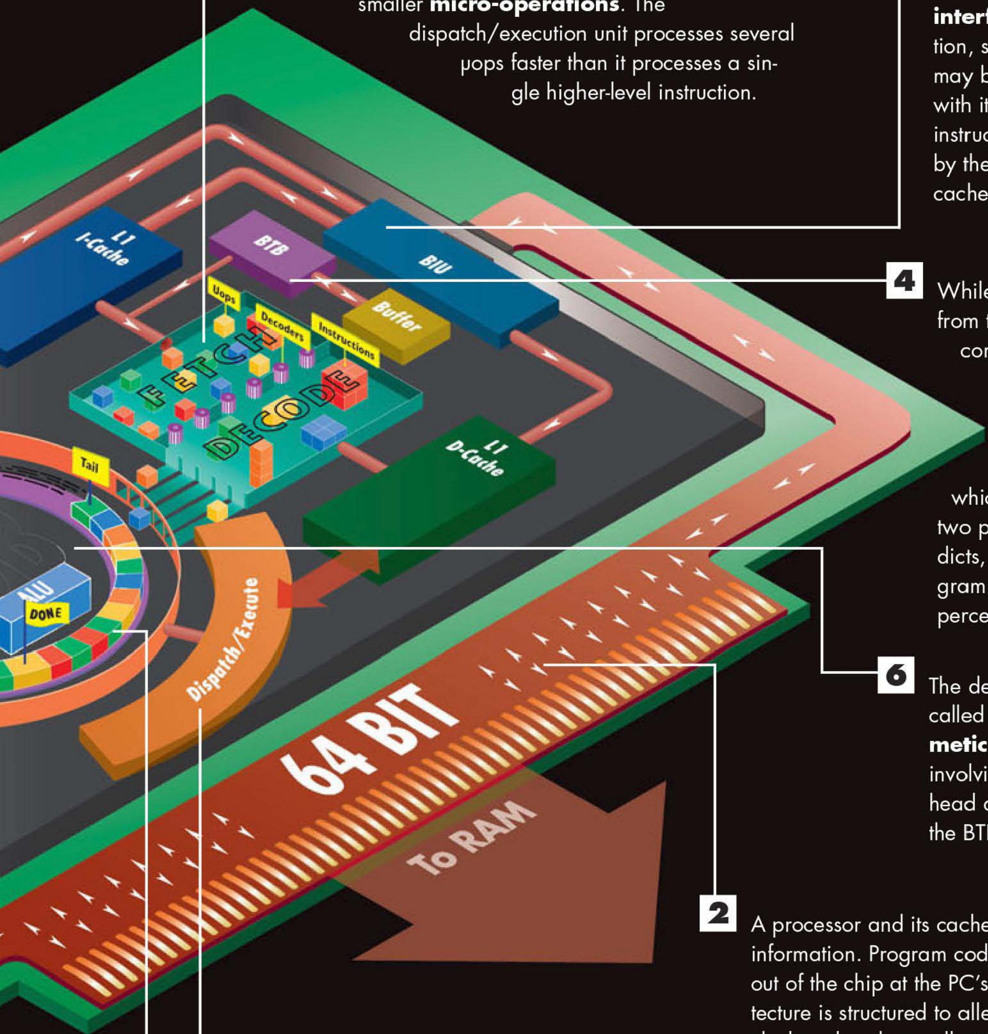
**11** When a µop that had been delayed is finally processed, the execute unit compares the results with those predicted by the BTB. Where the prediction fails, a component called the **jump execution unit (JEU)** moves the end marker from the last µop in line to the µop that was predicted incorrectly. This signals that all µops behind the end marker should be ignored and can be overwritten by new µops. The BTB is told that its prediction was incorrect, and that information becomes part of its future predictions.

**8** If a µop needs data from memory, the execute unit skips it, and the processor looks for the information first in the nearby L1 cache. If the data isn't there, the processor checks the much larger L2 cache. Because the L2 cache is integrated with the CPU, information moves between them faster than between the CPU and the external bus.

**10** If an operation involves floating-point numbers, such as 3.14 or .33333, the ALUs hand off the job to the floating-point math unit, which contains processing tools designed to manipulate floating-point numbers quickly.

**5** As the fetch/decode unit pulls instructions in the order predicted by the BTB, three decoders working in parallel break up the more complex instructions into **μops**, which are smaller **micro-operations**. The dispatch/execution unit processes several μops faster than it processes a single higher-level instruction.

**3** When information enters the processor through the **bus interface unit (BIU)**, the BIU duplicates the information, sends one copy to the CPU's closest caches, which may be built directly into the CPU's **die** or is housed with it. The BIU sends program code to the Level 1 instruction cache, or I-cache, and sends data to be used by the code to another L1 cache, the data cache (D-cache).




**4** While the fetch/decode unit is pulling in instructions from the I-cache, the **branch target buffer (BTB)** compares each instruction with a record in a separate set-aside buffer to see whether any instruction has been used before. The BTB is looking in particular for instructions that involve **branching**, a situation in which the program's execution could follow one of two paths. If the BTB finds a branch instruction, it predicts, based on past experience, which path the program will take. The predictions are better than 90 percent accurate.

**6** The decode unit sends all μops to the instruction pool, also called the **reorder buffer**. This contains two **arithmetic logic units (ALUs)** that handle all calculations involving integers. The ALUs use a circular buffer, with a head and tail, that contains the μops in the order in which the BTB predicted they would be needed.

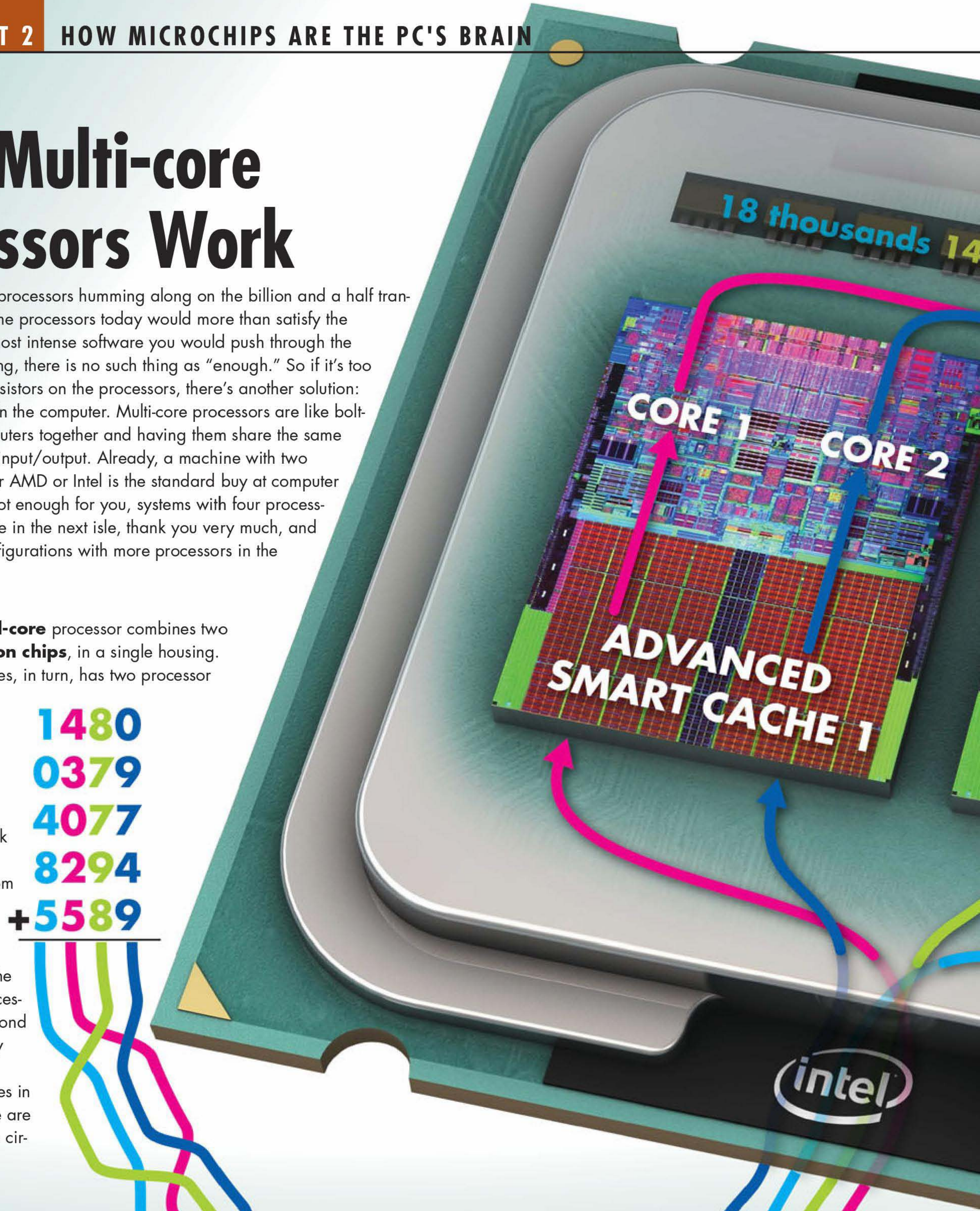
**2** A processor and its cache share the same 64-bit interface to the computer's information. Program code or data manipulated by that code move in and out of the chip at the PC's maximum bus speed. Much of a computer's architecture is structured to alleviate the bus bottleneck by minimizing the times a clock cycle—the smallest time in which a computer can do anything—ticks away without the processor completing an operation.

**7** The dispatch/execute unit checks each μop in the buffer to see whether it has all the information needed to process it, and when it finds a μop ready to process, the unit executes it, stores the result in the micro-op itself, and marks it as done.

**9** Instead of sitting idle while that information is fetched, the execute unit continues inspecting each μop in the buffer for those it can execute. This is called **speculative execution** because the order of μops in the circular buffer is based on the BTB's branch predictions. The unit executes up to five μops simultaneously. When the execution unit reaches the end of the buffer, it starts at the head again, rechecking all the μops to see whether any have finally received the data they need to be executed.

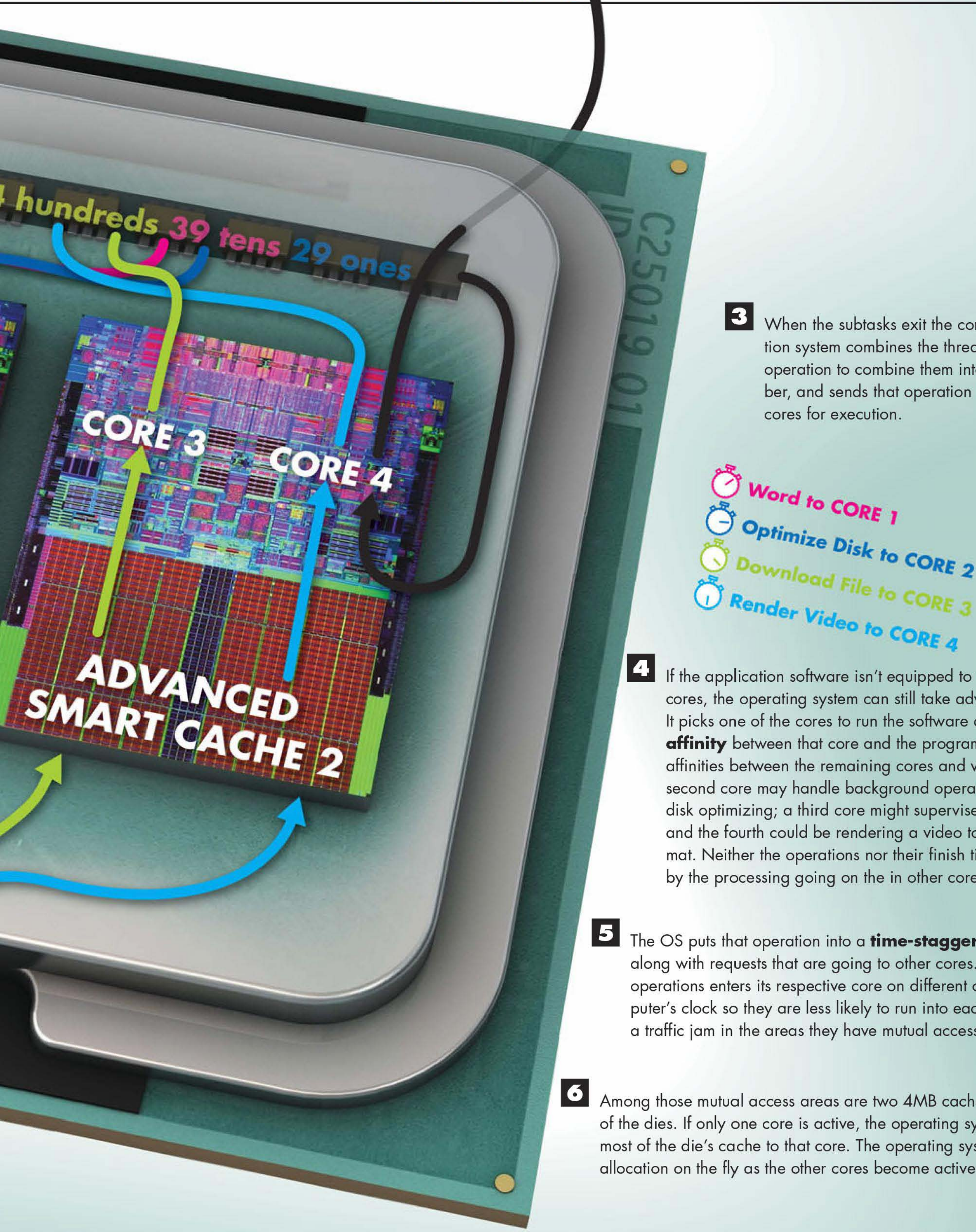
# How Multi-core Processors Work

You'd think the microprocessors humming along on the billion and a half transistors you find in some processors today would more than satisfy the requirements of the most intense software you would push through the chips. But in computing, there is no such thing as "enough." So if it's too hard to put more transistors on the processors, there's another solution: Put more processors in the computer. Multi-core processors are like bolting a couple of computers together and having them share the same memory, power and input/output. Already, a machine with two processors from either AMD or Intel is the standard buy at computer stores. And if that's not enough for you, systems with four processing units are available in the next isle, thank you very much, and Intel already has configurations with more processors in the works.




**1** An Intel **quad-core** processor combines two **dies**, or **silicon chips**, in a single housing. Each of the dies, in turn, has two processor **execution cores:** the heart of any microprocessor and the part that does the heavy work of executing instructions from software. The two cores are twins to each other and to the two other processors in the second die. The wildly colored areas above the cores in the photo here are supporting the circuitry.

**2** To gain the speed and other advantages possible with a multiprocessor computer, the **operating system (OS)** running on it must be designed, as are Windows XP and Vista, to recognize that a PC has multi-core processors, be able to distinguish them, and know how to handle operations such as **hyper-threading**. Similarly, software applications, games, and utilities need to be rewritten to use the multiple cores. Such software is referred to as **threaded**, or **multi-threaded**. The software adding, say, a column of four-place numbers could divide the job into four threads: Adding the 1s-place numbers, the 2s-place numbers, the 3s-place numbers, and the 4s-place numbers. Each of those subtasks is directed to a different core.




**3** When the subtasks exit the cores, the operation system combines the threads into an operation to combine them into a single number, and sends that operation to one of the cores for execution.

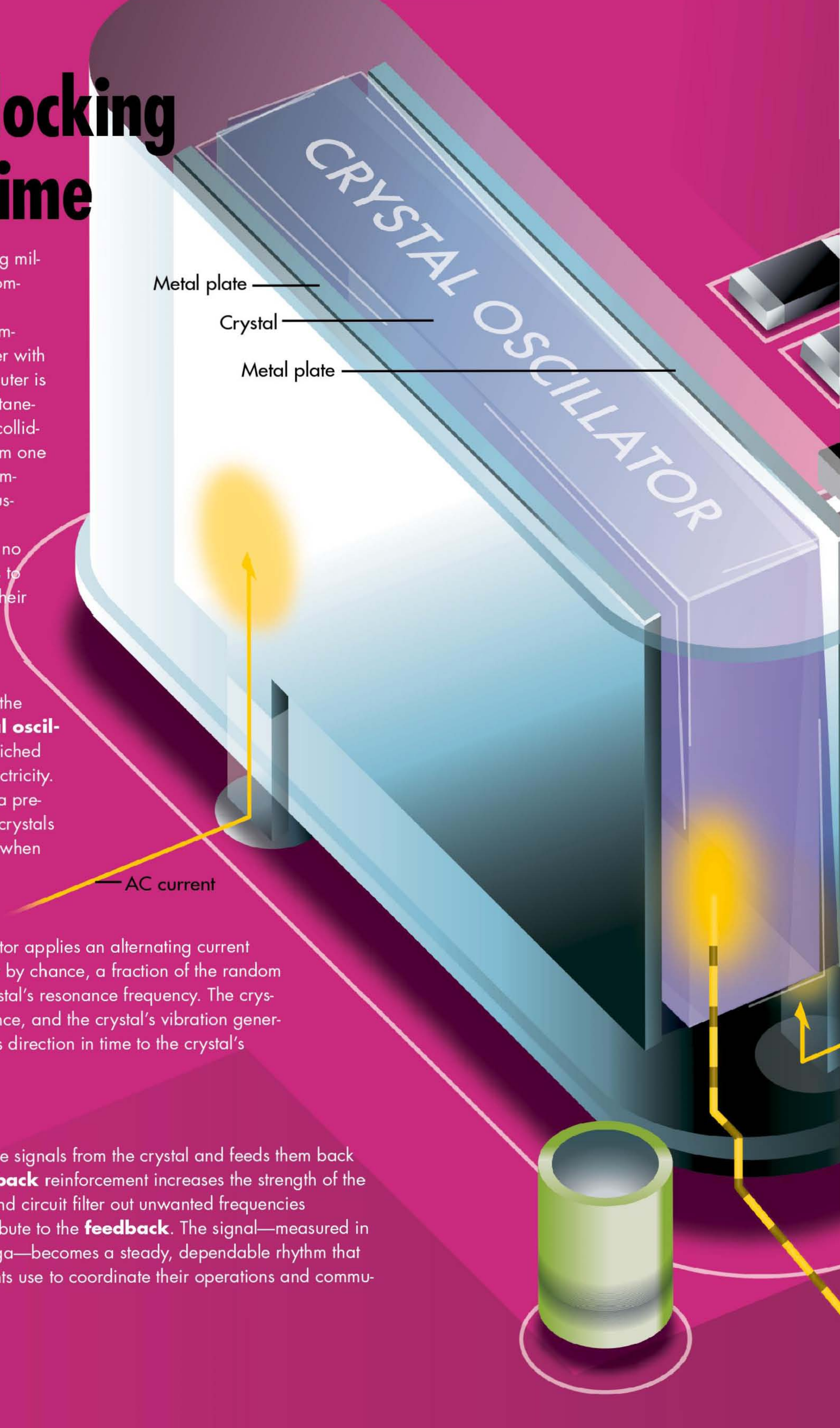
**4** If the application software isn't equipped to work in multiple cores, the operating system can still take advantage of them. It picks one of the cores to run the software and creates an **affinity** between that core and the program. It then creates affinities between the remaining cores and various tasks. A second core may handle background operations, such as disk optimizing; a third core might supervise a download; and the fourth could be rendering a video to a different format. Neither the operations nor their finish times are affected by the processing going on the in other cores.

**5** The OS puts that operation into a **time-staggered queue** along with requests that are going to other cores. Each of the operations enters its respective core on different clicks of the computer's clock so they are less likely to run into each other or cause a traffic jam in the areas they have mutual access to.

**6** Among those mutual access areas are two 4MB caches, one for each of the dies. If only one core is active, the operating system devotes most of the die's cache to that core. The operating system changes that allocation on the fly as the other cores become active or inactive.

# How Overclocking Multiplies Time

Every computer has an internal clock ticking millions of times a second. That clock is the computer's metronome, conductor, and choreographer. It assures that data and commands pass from one component to another with deft, precise timing. With its clock, a computer is like a troupe of a half-dozen jugglers simultaneously tossing heavy clubs that barely miss colliding with each other on their trajectories from one juggler to the next. Without the clock, a computer is like a troupe of jugglers with concussions and broken bones. But computer **modders**—born experimenters for whom no off-the-shelf PC is fast enough—found ways to raise a PC's heart rate without destroying their PCs. Well, usually.




**1** All that a computer does, it does to the beat of its own drummer—a **crystal oscillator** made of quartz that is sandwiched between two plates that conduct electricity. Quartz has a natural **resonance**, a predictable frequency at which quartz crystals of the same size and shape vibrate when electricity passes through them.

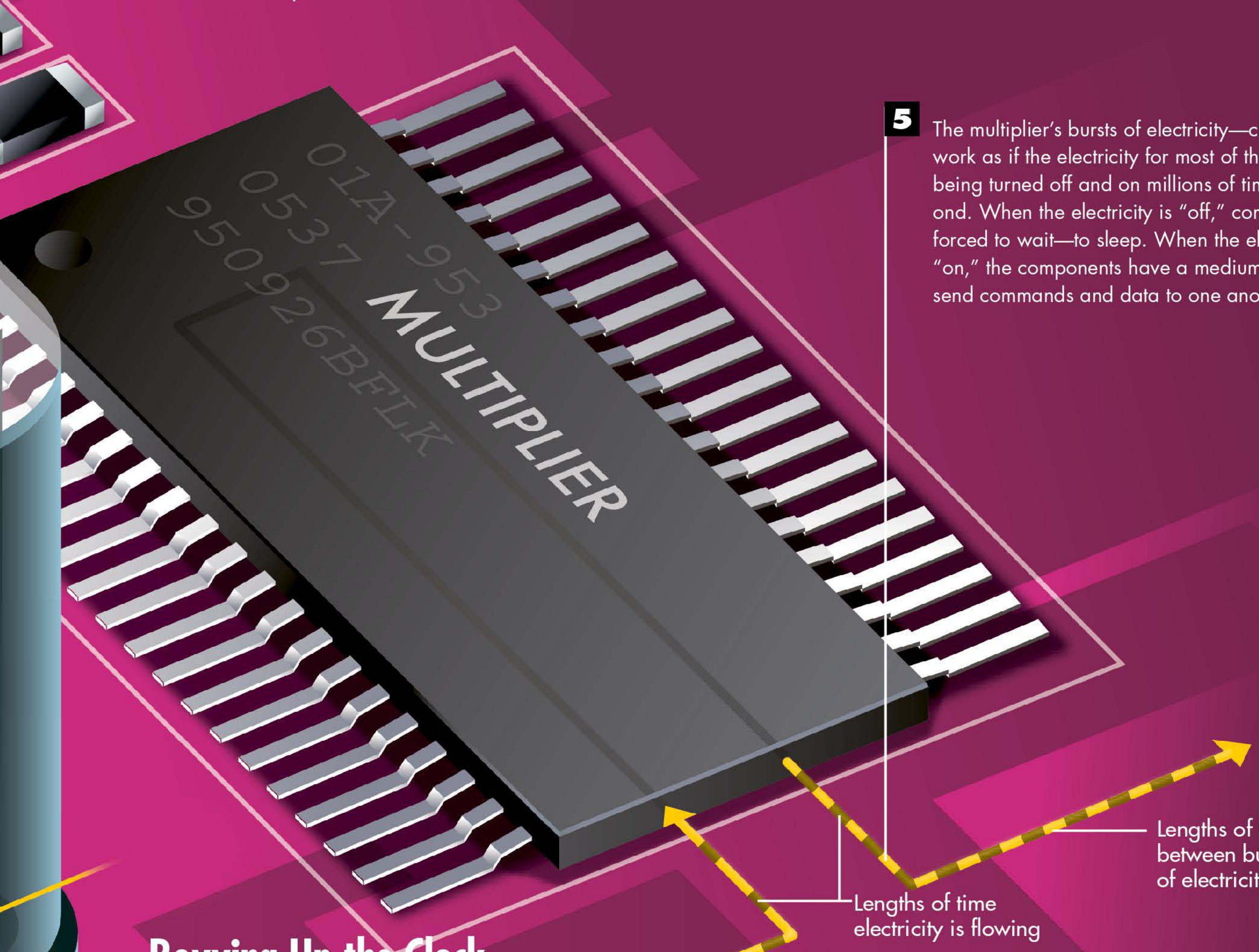
**2** During startup, a circuit in the oscillator applies an alternating current (AC) signal to the crystal, and purely by chance, a fraction of the random noise in the current will be at the crystal's resonance frequency. The crystal locks onto its own natural resonance, and the crystal's vibration generates another AC current that switches direction in time to the crystal's vibrating.

**3** The oscillator amplifies the signals from the crystal and feeds them back to the crystals. The **loopback** reinforcement increases the strength of the signal while the crystal and circuit filter out unwanted frequencies because they don't contribute to the **feedback**. The signal—measured in hertz, both mega and giga—becomes a steady, dependable rhythm that the computer's components use to coordinate their operations and communications.

**4** Few computers today have clock rates based solely on the frequency of the oscillating crystal. It's too slow. Instead, a circuit called a **multiplier** manipulates the clock's **clicks**, or pulses of electricity, by opening and closing switches and slicing the clock's current to create an **overtone frequency** higher than the crystal's **fundamental resonance**.

Another factor in the speed of a PC is the **frontside bus**—the circuitry that connects the CPU, RAM, and memory controller. The bus has an independent clock that affects the overall speed of the computer. The simple formula for determining the clock speed of a PC is: Frontside Bus (Hz) × CPU Multiplier × Crystal's Resonance = Operating Frequency (Hz)

**5** The multiplier's bursts of electricity—clock **ticks**—work as if the electricity for most of the PC were being turned off and on millions of times a second. When the electricity is "off," components are forced to wait—to sleep. When the electricity is "on," the components have a medium by which to send commands and data to one another.




## Revving Up the Clock

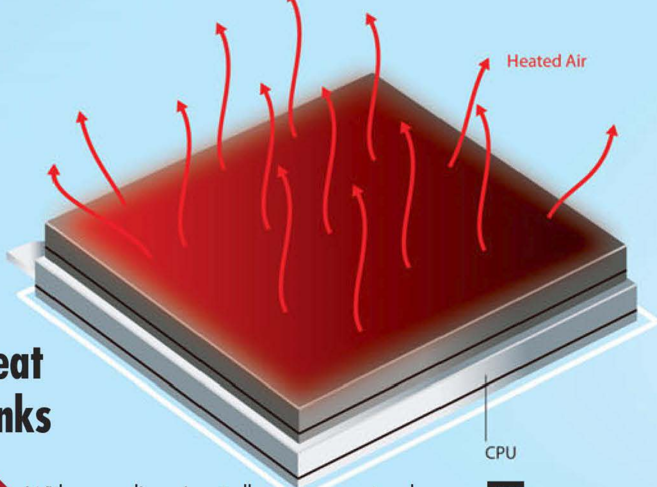
The more often the electricity is "on," the more opportunities components have to get something accomplished. This has led to the practice of **overclocking**— manipulating parts so there are more clicks per second—which takes advantage of the fact that chip makers leave considerable margin about the rated frequency of their processors to maintain the microchip's reliability. There are basically two ways to overclock: increase the multiplier rate or increase the frontside bus speed.

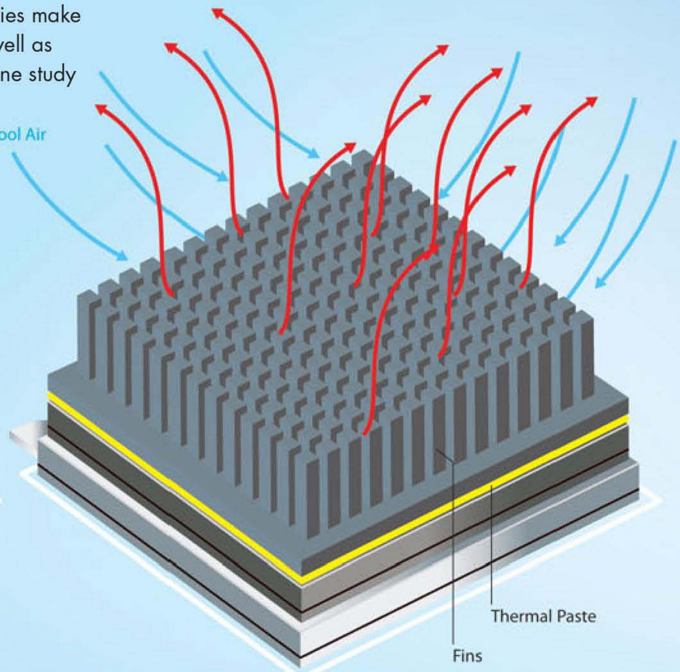
Increasing the multiplier rate used to be simple to do, either by setting jumpers or changing values in the motherboard's setup screen. But chip makers, for whom overclocking has no appeal, have locked multiplier rates. It's still possible to inflate the multiplier rate by adding traces between circuits, sometimes with only a pencil line. But it's tricky without thorough knowledge of what the circuits do.

The overclock method that remains is resetting the frontside bus speed. For this, you need a motherboard by a company that supports overclocking. You make changes using controls thoughtfully provided in the set-up screens. You should make changes in small increments and run your PC for a while before going further to avoid *over*-overclocking, which results in overheating, sometime fatal to your PC.

# How a PC Keeps Its Cool

Overclocked PCs are invariably overcooked PCs. The collisions of electrons moving through the traces and wires of microchips and circuit boards generate heat. The heat, in turn, contributes to the electrons' volatility. Like speeding cars, fast-moving electrons can lose the ability to stay on the track, especially as PC companies make circuits —their tracks— narrower. The result is that heat causes errors, as well as faster deterioration of the materials that make up computer components. One study found that every increase of 10° Celsius (C) drops the reliability of PC components by 50 percent. Aggressive computer users and makers have found several ways to cool down their machines.

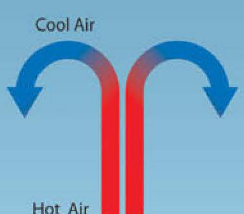



## Heat Sinks

**1** With no cooling airs at all, a component such as the CPU transfers its heat by **conduction**. Hot, rapidly vibrating molecules within the chip bump against neighboring atoms and molecules, transferring heat in the process. Eventually the heat reaches the chip's surface, where conduction transfers the heat to the air, allowing the heat to slowly dissipate.

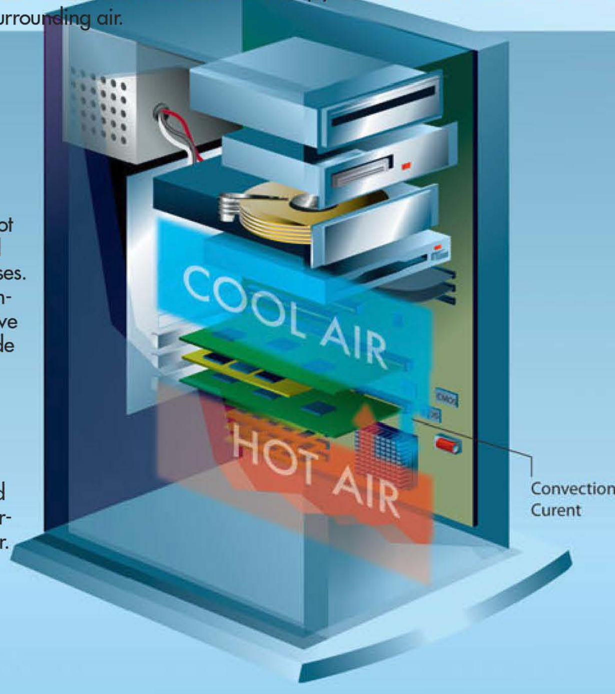
**2** The more surface area a chip has, the more heat can be passed on to the air. The added surface is provided by **heat sinks**. The sink is glued to a chip with a highly conductive **thermal paste** that assures there are no gaps filled with the less conductive air. Heat passes into the sinks, usually made of copper or aluminum, which rapidly move the heat to dozens of thin **fins** that multiply the surface area to transfer more heat to the surrounding air.



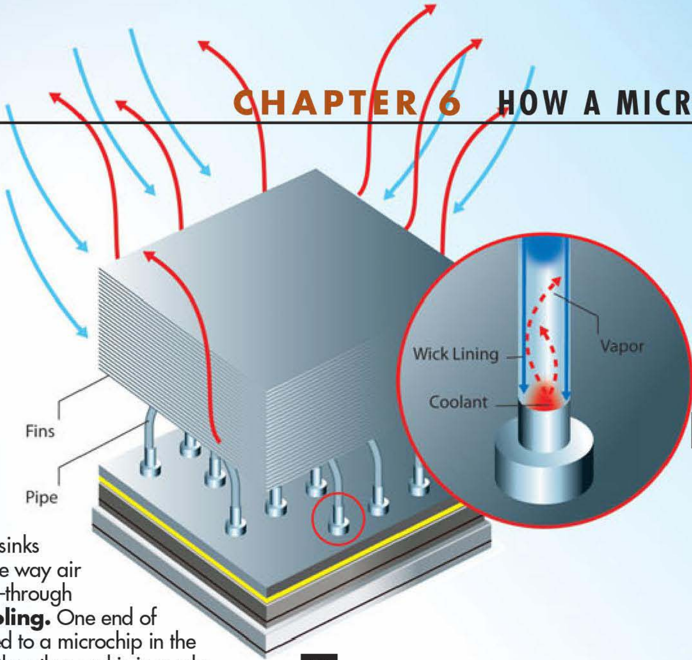

## Forced Convection

**2** The lighter hot air rises, and the denser and heavier cool air falls. Away from the heat source, the hot air cools and sinks; the fallen cool air is heated by the source and rises. The result is a roughly circular convection system. But the cables, drive bays, and expansion boards inside a computer block the circulation, making convection less efficient.

**1** Even with heat sinks and heat pipes, there remains the problem of getting the heat transferred to the air, away from the sinks and pipes, so they can pass more heat into the air. This is performed naturally through **convection**, based on the fact that the more rapidly moving atoms and molecules in hot air are farther apart from each other, making the hot air less dense and, therefore, lighter.



# Heat Pipes

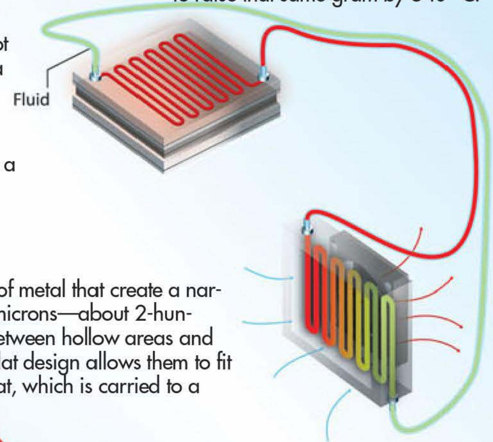



**1** **Heat pipes** are supercharged heat sinks that cool in the same way air conditioning does—through **evaporative cooling.** One end of the pipe is connected to a microchip in the way a heat sink is; the other end is in a relatively cool part of the computer. The pipe is a sealed, hollow tube that contains a small amount of fluid coolant, usually some combination of ammonia, alcohol, and water. The rest of the tube contains a near vacuum with only a thin vapor of the coolant.

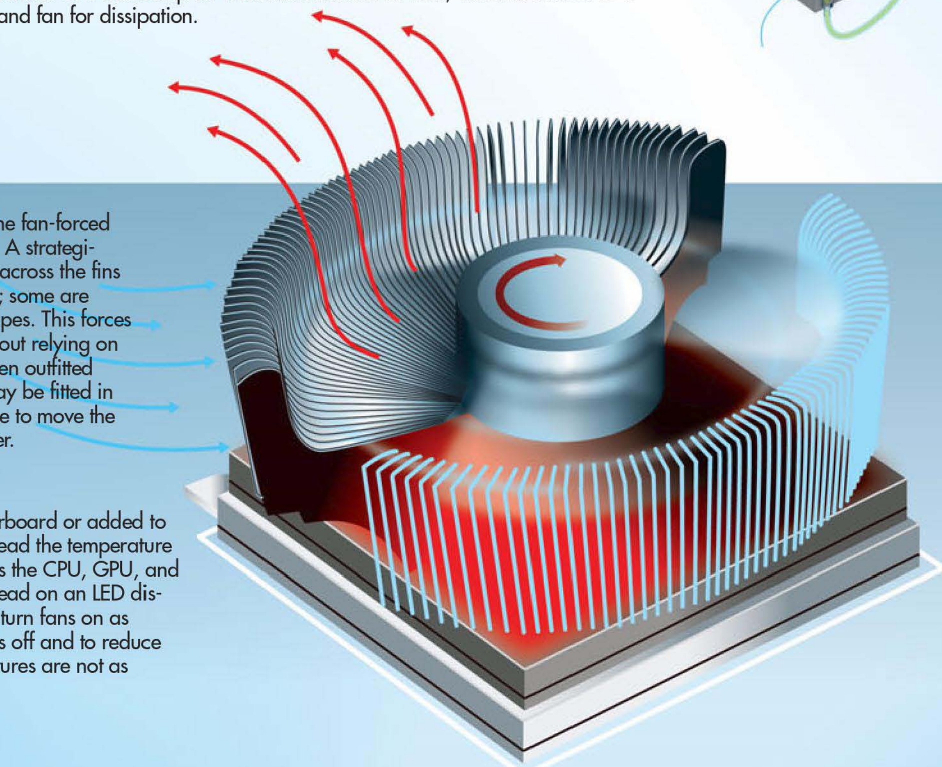
**2** Heat from the microchip evaporates the coolant, drawing heat from the hot end of the pipe through a process called **latent heat,** which occurs whenever a substance changes its state, such as a liquid becoming a gas.

**3** The evaporation increases vapor pressure inside the pipe at the end touching the chip. This results in a rush of vapor toward the cool end of the pipe as the pressure seeks to equalize itself throughout the pipe. At the cool end, the vapor condenses, releasing the heat it carried with it from the hot end. The cool end is typically enveloped by fins that dissipate the heat to the surrounding air. Evaporation of a single gram of water uses the energy required to raise that same gram by 540° C.

**4** The liquid resulting from condensation returns to the hot end, either through a **wick** that coats the inside of the pipe or simply by running down the inside of pipes mounted vertically.




**5** Not all heat pipes are pipes. **Flat heat pipes** are two thin sheets of metal that create a narrow cavity between them. Some flat pipes are no thicker than 500 microns—about 2-hundredths of an inch. The space between the metal sheets is divided between hollow areas and sheets of a thin material with capillaries to act as wicks. The pipes' flat design allows them to fit flush against the surface of a microchip for maximum transfer of heat, which is carried to a remote heat sink and fan for dissipation.

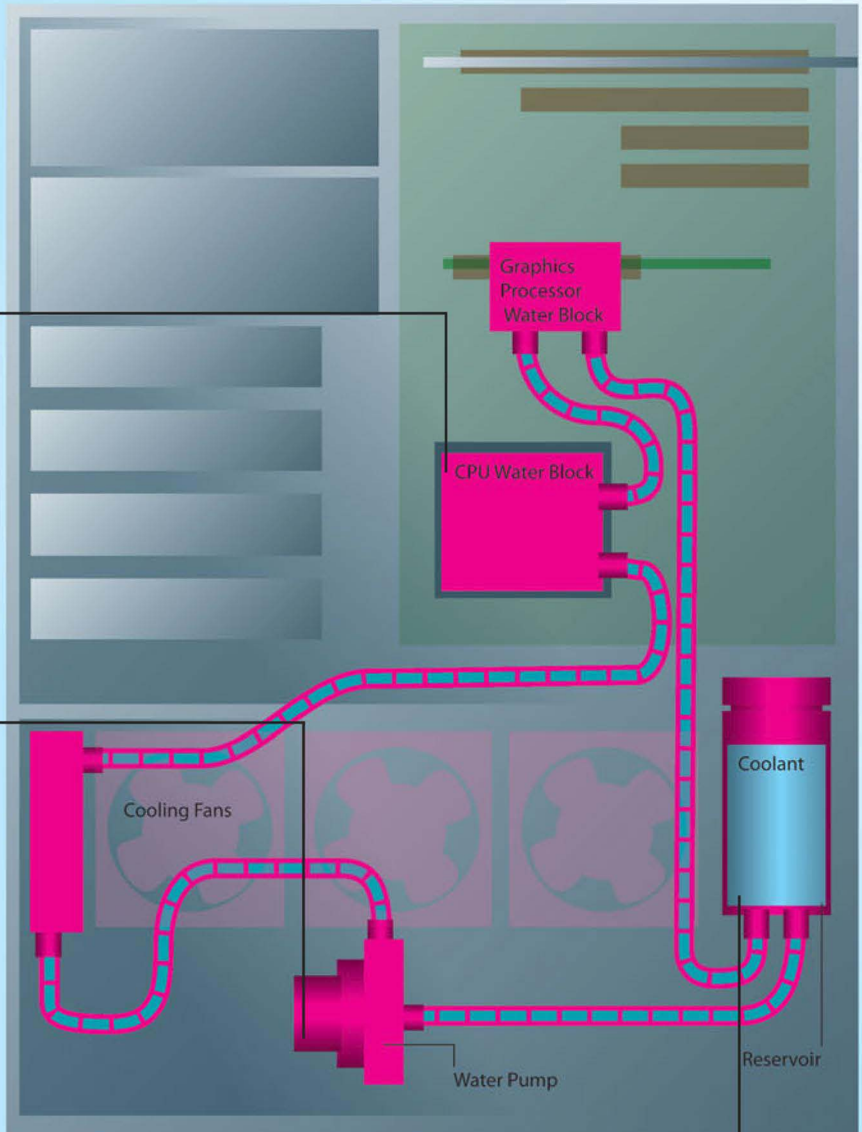


**3** One solution is similar to the fan-forced convection found in ovens. A strategically-placed fan blows air across the fins of a heat sink or heat pipe; some are integral parts of sinks or pipes. This forces the hot air to circulate without relying on convection. Other fans, often outfitted with neon or LED lights, may be fitted in holes in the computer's case to move the hot air outside the computer.

**4** Probes built into the motherboard or added to the PC by an overclocker read the temperature of key components, such as the CPU, GPU, and RAM. The results may be read on an LED display or may automatically turn fans on as needed to cool components off and to reduce noise levels when temperatures are not as great.

# How Advanced Cooling Solutions Refrigerate Your PC

## Water Cooled PCs



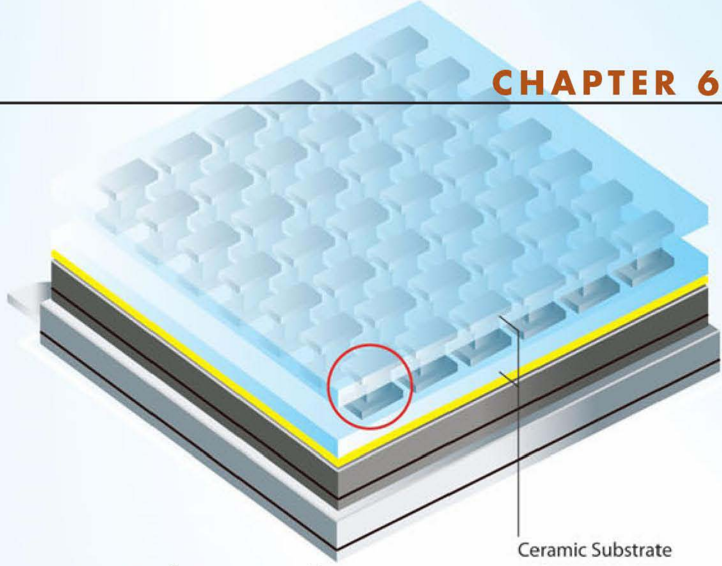

1 The most popular solution to eliminating the noise of fans while at the same time providing an even lower temperature is the same as that found in cars and the Browning .50-caliber machine gun: water cooling.

2 The same type of thermal paste used with heat sinks holds a **water block** tightly to the CPU. Inside the block is a watertight channel that travels back and forth to create the maximum opportunity for heat from the CPU to transfer to a fluid inside the channel. Similar water blocks may also be attached to the North Bridge and South Bridge chips and the graphics processing unit on the video card—all sources of extreme heat.

3 If there are more than one microchips being cooled, they are daisy-chained by tough plastic tubing designed to take turns without kinking. A pump continually moves liquid through the tubing. The liquid might be distilled water, to which additives may be added to improve the water's ability to absorb heat, to discourage corrosion and bacterial growth, to give the fluid a bright color, or to make it glow under ultraviolet light.

4 The pump moves the hot coolant to a metal **radiator** located outside the computer, or at least an area far-removed from the hotter components. The radiator has an undulating channel that gives the metal ample opportunity to absorb heat from the water. The heat moves through the absorbing metal to a system of fins that, as on a heat sink, expands the surface area for the heat to dissipate into the air. There may or may not be a fan cooling the fins, depending on the efficiency of the water system and the owner's desire for a quiet system.

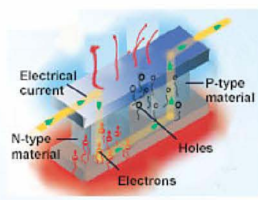
5 The cooled liquid flows through a **reservoir**. The reservoir provides another area where heat can escape. Only part of the reservoir is filled with liquid. Air in the reservoir allows the liquid to expand and contract as its temperature changes without putting undue pressure on the tubing. It also provides a convenient place for adding additional coolant if needed. From the reservoir, the cooled coolant returns to the system of water blocks to continue the cooling process.




# Peltier Cooling

**1** One of the more exotic methods of cooling computers uses the **Peltier effect**, first observed by Jean Peltier in 1834. The effect appears in a device made of two dissimilar metals—metals that react differently to electrical current—that are joined in at least two different places. A Peltier device, also called a **thermo-electric cooler**, can also work with two different types of semi-conductors—**n-type**, to which impurities have been added so it has extra electrons, and **p-type**, in which impurities create **holes**, or vacancies where electrons could be. For simplicity, we'll just use semiconductors for our example.

**2** When electricity flows through the n-type elements and crosses through a metallic con-nector to the p-type material, the extra electrons in the n-type semiconductor flow in the opposite direction of the elec-trical current.



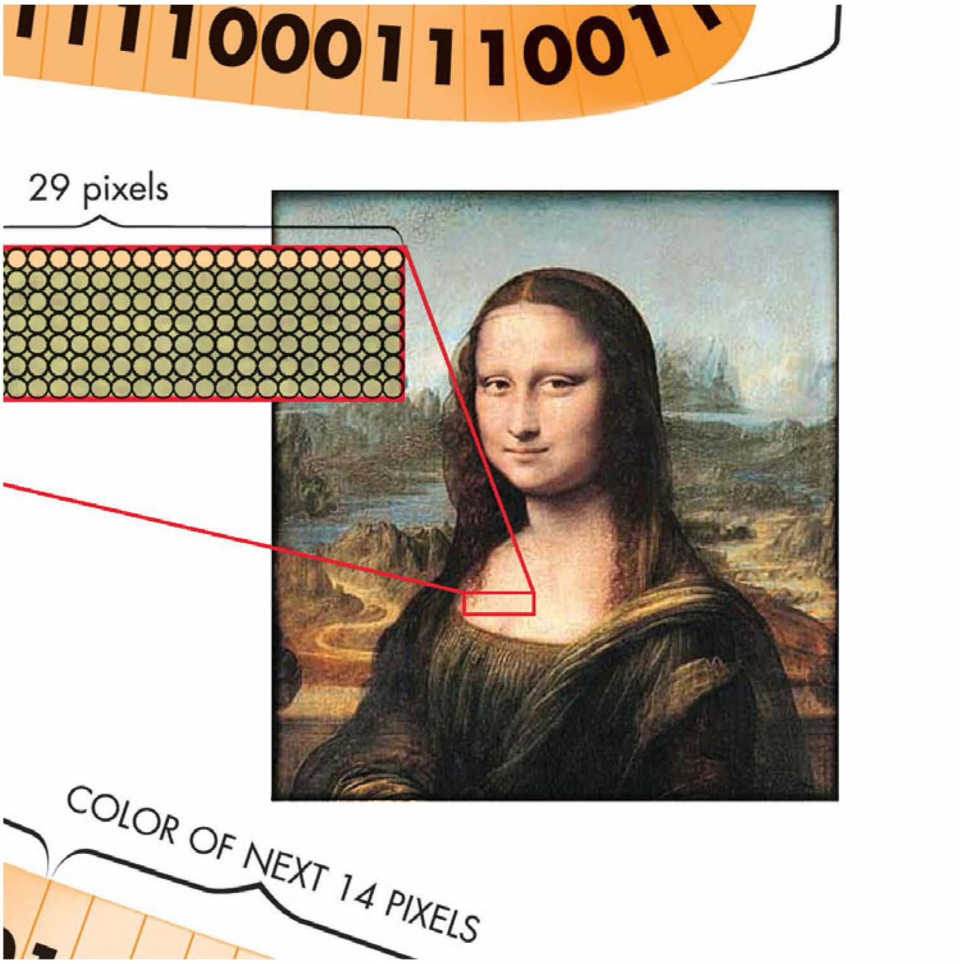

**3** After the current continues on and passes through the p-type material, it returns to the n-type through another metallic bridge. This time the holes in the p-type element move through the metal bridge in the same direction as the current.

**4** In both instances, the electrons and the holes carry heat with them. The junction they are headed for warms, and the junction they are leaving cools off. The cool side of the device is attached to a microchip like a heat sink, and the hot side uses some combination of fins, heat pipes, and fans to dissipate the steady stream of heat originating at the chip. (Despite its high-tech nature, a Peltier device is an electricity hog and only 49-60 percent as effective as a refrigerator compressor. )

### Cooling with Crisco

The strangest method overclockers have developed to keep a computer cool is to submerge the motherboard and its expansion boards in cooking oil. Oil has dual strengths—it's an insulator and an excellent conductor of heat, making it a good, if messy, substance for cooling PCs, as well as frying chicken.

***Photo courtesy of Tom's Hardware***




**1614**
John Napier builds Napier's Bones, rods that are used as mechanical aids to calculation.

**1621**
William Oughtred invents the slide rule, which does not become obsolete for nearly 350 years, when it's supplanted by the pocket calculator in 1970.

**1679**
Gottfried Leibniz introduces binary arithmetic, showing that every number can be represented by the symbols 0 and 1 only.

**1804**
Joseph-Marie Jacquard programs a weaving loom that uses a series of punched cards to form patterns in the woven fabric. This inspired a series of computers (even well into the 20th century) that received their data from punched cards or tape.

**1854**
George Boole develops a system of mathematics called Boolean algebra, which uses binary operations.

**1854**
Augustus DeMorgan, in conjunction with Boole, formalized a set of logical operations now known as DeMorgan transformations.

**1890**
The U.S. Census Bureau faces its first data-processing crisis due to surging population growth. Herman Hollerith's electric tabulating system helps them solve the numeric nightmare. Before the adoption of this punched-card system, the census took eight years to complete; with Hollerith's machine, it takes just two years. Hollerith goes on to form the Tabulating Machine Company in 1896, which later merges with two other companies to become International Business Machines Corporation, popularly known as IBM.

P A R T

# 3

# How Software Works

C H A P T E R S



**1903**
Nikola Tesla, a Yugoslavian scientist/inventor, patents electrical logic circuits called "gates" or "switches."

**1904**
John Ambrose Fleming experiments with Edison's diode vacuum tube (an invention Edison did not pursue), developing the first practical radio tubes. The vacuum tube's application to computers was not exploited until the late 1930s.

**1936**
Konrad Zuse, a German, creates a programmable, digital computing machine that introduces important concepts such as the use of the binary system and valves.

**1937**
Alan Turing invents the Turing Test, which determines whether something is human. Questions include things such as "Do you have feelings?" and "Do you feel pain?"

**1943**
The U.S. Army Ordnance Department commissions the ENIAC (Electronic Numerical Integrator Analyzer and Computer) to produce missile trajectory tables for World War II. ENIAC can perform 5,000 additions per second and later is used in artillery calculations. ENIAC weighs 30 tons and is 100 feet long, 8 feet high, and contains 17,468 vacuum tubes. Programming ENIAC is anything but user friendly; it takes two days to set up problems that ENIAC solves in two seconds.

**1945**
John von Neumann writes the first draft of a report on the EDVAC (Electronic Discrete Variable Computer), in which he describes a general purpose electronic digital computer with a stored program. It does not get built until 1952.

***Sa-la-ga-doo-la Men-chic-ka Boo-la Bibbidi-Bobbidi-Boo. Put them together and what have you got? Bibbidi-Bobbidi-Boo.***

—Walt Disney's *Cinderella*

**THE** language programmers use to create software sound a lot like the fairy godmother's incantation: grep, mov, endif, cur_x, and selField. And software really is magical. You slide a DVD into your PC, or you invoke your PC—maybe with a touch, maybe at the sound of your voice—or you point at a little picture with a mouse and click, and suddenly all these things begin to happen that could only be witchery. Beautiful color images and voices and sounds emanate from your PC. The software looks at a few numbers and predicts banana futures in three months. Ask for information on a person, a country, or a date, and the software responds like a crystal ball. You ask the software to take you to another computer on another continent, and in seconds you're there; magic carpets really can't compete.

Software's abracadabra needn't be a mystery to those of us who don't speak BASIC, Java, Pascal, C, C++, C#, Lisp, and other curiously named software languages. Even if you've never touched a computer before, you've used software. Software doesn't include only things like WordPerfect and Microsoft Excel. Software is also music recordings and videotapes. A recipe for Aunt Hattie's chocolate pie, a dress pattern, and a telephone number are programs. A program is simply any set of instructions for an ordered series of actions. A program can exist as a printout of computer code or as a recipe in a cookbook. And you're already a programmer, even if you've never touched a computer.

Have you played the latest Stones' CD? That's software! Can you program your VCR? You're creating software. A computer software program—not really redundant, trust me—is simply a recipe, no less, no more than a recipe for Aunt Hattie's chocolate pie.

Think of what you do when you use a microwave. You press buttons in a certain order to make the microwave work at a specific power level for a certain length of time, then change to a different power level, and so on until your program produces beef stew. That's programming—a set of instructions in a particular order, created on the fly. But if you press a button that's preset for microwave popcorn, you're using software—a preconfigured set of programming instructions that, in this case, are recorded permanently in a microchip inside the microwave.

**1945**
First computer "bug" reported. The "bug" was a moth that was caught up in the computer. It was discovered by naval officer and mathematician Grace Murray Hopper.

**1948**
Britain's Manchester Mark I is the first computer that can store a program electronically rather that requiring programmers to set switches manually.

**1951**
Although promised for delivery in 1948, UNIVAC is delivered to the U.S. Census Bureau three years late. It's a sensation, nevertheless, with revolutionary features such as mercury delay lines for memory and magnetic tape for input instead of punched paper.

**1956**
At MIT, researchers begin experimentation on direct keyboard input on computers.

**1960**
A team drawn from several computer manufacturers and the Pentagon develops COBOL, Common Business Oriented Language.

**1960**
Digital Equipment Corporation introduces its first minicomputer, the PDP-1, priced at a relatively modest $120,000.

**1960**
ERMA (Electronic Recording Method of Accounting) replaces 2,332 bookkeepers at the Bank of America.

**1960**
LISP makes its debut as the first computer language designed for writing artificial intelligence programs.

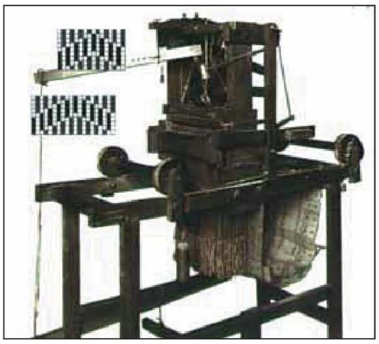
**1962**
Werner Frank co-founds Informatics, the company that sells Mark IV, the first million-dollar software product, in 1962.

**1962**
MIT students Slug Russell, Shag Graetz, and Alan Kotok write *Spacewar*, considered the first interactive computer game.

Hardware is capable of doing a variety of tasks, but without programming and software to control it, hardware can't do much of anything. By itself, for example, a claw hammer is useful only as a paperweight, but used by a carpenter, it can drive nails, pull out old nails, and even crack walnuts. The carpenter swinging the hammer is programming it on the fly. A home entertainment center is a complex system of hardware capable of generating a variety of sights and sounds, but it can't whisper a peep on its own. It needs software in the form of a compact disc, videotape, or at least a basic cable connection. The signals produced by the disc, tape, and broadcasts tell the hardware what electrical pulses re-create the sounds of Mahler's Tenth Symphony or the sights of *Terminator 2*.

## Software of the Second Millennium

If you think of software as a recorded set of instructions that control the actions of a machine, then software's really been around for quite some time. Music notation is written instructions for programming a piano on the fly. But if you have a player piano, the rolls of punched paper are its software. In the 18th century, weaving was done by someone manually slipping a bobbin wrapped with thread over and under other threads on a loom. The process was slow and rife with opportunities for slip-ups. But in 1804, Joseph-Marie Jacquard programmed a weaving loom using a series of punched cards that controlled what patterns were woven into a fabric. Different cards resulted in different patterns. Jacquard's invention is considered the birth of modern computer programming, and punch cards were used with computers well into the 20th century. The invention was also the occasion for the first wave of automation fear. In 1811, Ned Ludd, a Nottingham weaver who feared the new looms would replace people, led his co-workers in a frenzied attack on the machinery. *Luddite* has come to mean any person who resists technology advancements.



**The Jacquard Loom**
In 1804, Joseph-Marie Jacquard programmed a weaving loom using a series of punched cards to form patterns in the fabric. Punch cards to enter data and instructions into computers were used well into the 20th century.

Software made little progress for the next century. In 1889, Thomas A. Edison invented the kinetoscope, a hardware device that let people view moving pictures on

**1962**
Ivan Sutherland, for his MIT doctoral thesis, creates and demonstrates a graphics system called Sketchpad. It's the first program to use windows, icons, and a light pen, allowing easy manipulation of graphics and text onscreen.

**1963**
Finalization of the ASCII code (American Standard Code for Information Interchange) permits machines from different manufacturers to exchange data.

**1964**
John Kemeny and Thomas Kurtz develop the BASIC programming language at Dartmouth College. BASIC is an acronym for Beginners All-Purpose Symbolic Instruction Code.

**1967**
Seymour Papert designs LOGO as a computer language for children.

**1969**
Ken Thompson, a researcher at Bell Labs, writes the first version of UNIX, a multiuser, multitasking operating system. The UNIX source code is distributed freely throughout the 70s, and it soon becomes popular at universities and research labs.

**1969**
Bill Gates and Paul Allen, calling themselves the "Lakeside Programming Group," sign an agreement with Computer Center Corporation to report bugs in PDP-10 software in exchange for computer time.

**1975**
The MITS (Micro Instrumentation and Telemetry Systems) Altair 8800 appears on the cover of Popular Electronics. The article inspires Paul Allen and Bill Gates to develop a BASIC interpreter for the Altair.

**1976**
Bill Gates accuses hobbyists of stealing software and thus preventing "...good software from being written."

film. Film, sound recordings, and radio broadcasts—things we don't normally think of as software—were, in fact, just that, and the most prevalent form of software in the first half of the 20th century.

The first computers had neither a keyboard nor a monitor, and they had no software. Their creators programmed instructions by tediously flipping a series of switches in a precise arrangement. This set up a pattern of on and off electrical currents the computer then used to activate more electrical switches in the form of vacuum tubes. Finally, the result was displayed in the form of a panel of lights turned off or on to represent the zeros and ones in the binary number system.

John von Neumann in 1945 first proposed the idea of a general-purpose electronic digital computer with a stored program. But the computer, the ENIAC, was not built until 1952. Meanwhile, scientists in England in 1948 created the Manchester Mark I, the first computer that could store a program electronically instead of making programmers set switches manually.

During the next two decades, computers gained such accoutrements as keyboards and displays. They used software in the form of punched cards, punched paper tape, and magnetic tape. All those forms of software were awkward and slow to use compared to modern software, but they made life much easier for early programmers.

Not that there were many programmers back then to have their lives made easier. Some of the earliest writers of software sprang from a model railroad club at MIT. The railroaders used telephone switches to control their complex system of tracks, crossings, and rail switches. In the railroad, they had created a simple and very specialized form of computer. They already thought in terms of switches, the perfect training for writing software.

Most of the software writing then was going on in the universities, military, and businesses that were big enough to afford the then room-filling computers, called mainframes. But regardless of where different programs originated, in the 1960s they had one thing in common: They would work only on a specific computer for a specialized purpose. A program that was written, for example, to handle Gizmo Corp.'s payroll was customized specifically to match that company's accounting practices and record keeping, and it would run only on a computer configured like the one at Gizmo Corp. If another company

**1976**
The trade name "Microsoft" is registered with the Office of the Secretary of the State of New Mexico.

**1976**
Gary Kildall develops CP/M, an operating system that dominates most of the early microcomputers.

**1977**
The U.S. government adopts IBM's data encryption standard, the key to unlocking coded messages, to protect confidentiality within its agencies.

**1978**
Seymour Rubinstein invents WordStar, the most popular, versatile word processor for the next several years.

**1979**
Harvard MBA candidate Daniel Bricklin and programmer Robert Frankston develop VisiCalc, the program that made a business machine of the personal computer, for the Apple II.

**1981**
MS-DOS introduced.

**1982**
Mitch Kapor develops the spreadsheet program Lotus 1-2-3, greatly stimulating sales of the IBM PC.

**1983**
The first computer virus appears.

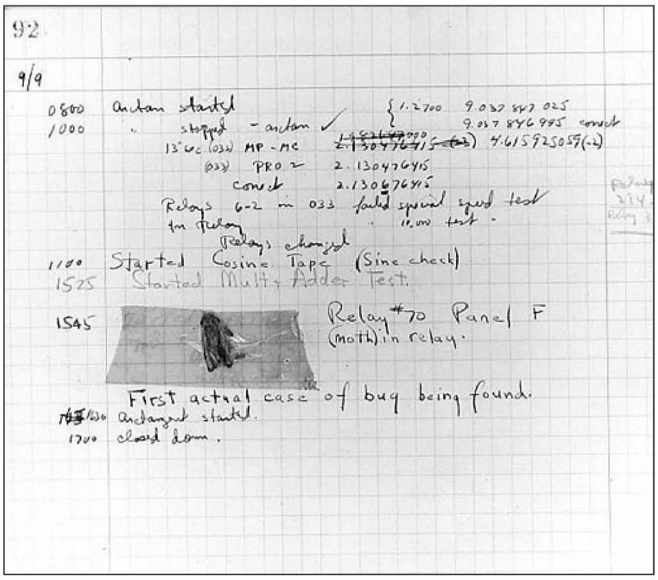
**1983**
Microsoft unveils Microsoft Windows, an extension of the MS-DOS operating system that provides a graphical operating environment.

**1984**
Satellite Software International introduces WordPerfect, a powerful new word processor for the IBM PC.

**1987**
Microsoft ships Microsoft Bookshelf on CD-ROM.

wanted a payroll program, it was written from the ground up again. And if a manager at *Gizmo* wanted to look at payroll information from a different angle, the entire program had to be altered.

Programs back then had one other thing in common: They were either very expensive or free. Computer companies, such as IBM, saw software as a tool to sell hardware—"big iron." Because each program was a custom job—in a discipline that was still finding itself—companies expected to spend thousands of dollars. And compared to a computer that cost millions, a few thousand for software didn't seem so bad.




Although a computer bug has come to be associated with faulty software, the bug from which the term derives its name was found in a computer's mechanical relay, causing it to malfunction. It was taped into a logbook by Dr. Grace Hopper.

On the other hand, computer companies didn't charge for an *operating system*—the crucial software that lets a computer run a program that performs specific applications. Programmers routinely swapped their software code with other programmers. Many programmers saw computers as more than business tools. Computers, somehow, were going to be the great equalizer. Information was going to be the next industrial revolution, and computers were going to give Joe Blow access to the same information available to the board of GM. Such idealism, joined with the fact that all programmers were really just in the learning stage, led to a philosophy that all software should be distributed for free. That ideal is still alive today in the open source movement, where powerful operating systems such as Linux are free and users are encouraged to make improvements for all to share.

Gradually, the nature of software changed. Computer companies began unbundling software from hardware. A new working class—the "cowboy," programmer for hire—traveled from company to company customizing programs and then moving on. In the mid-60s, it occurred to some of these cowboys that they could write one program and sell it to several companies. A new market was born.

**1988**
By only a small edge, Microsoft surpasses rival Lotus Development Corporation as the top software vendor.

**1989**
Maxis releases *SimCity*, a video game that helps launch simulation games as a new class of educational and entertainment software.

**1991**
Microsoft announces Microsoft Visual Basic for Windows.

**1994**
Microsoft announces Microsoft Windows 95.

**1995**
Microsoft ships Internet Explorer 2.0.

**2001**
Microsoft ships Windows XP. It's the first fully 32-bit operating system targeted for both business and home consumers.

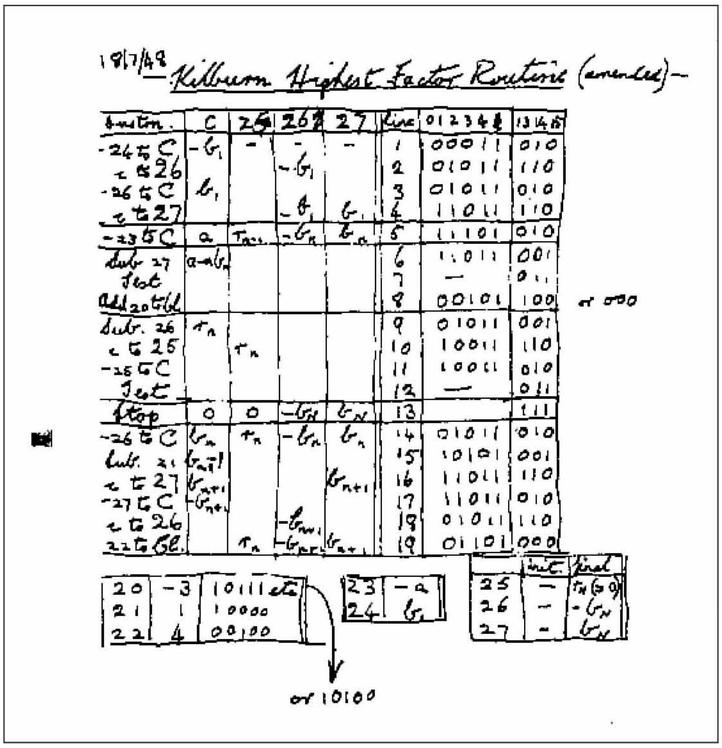
**1989**
A young student at the University of Helsinki, Linus Torvalds, releases a new UNIX variant, Linux.

**1990**
Microsoft ships Windows 3.0. Compatible with DOS programs, the first successful version of Windows finally offering good enough performance to satisfy PC users.

**1993**
Microsoft reports that the number of licensed users of Microsoft Windows now totals more than 25 million, making it the most popular graphical operating system in the world.

**1995**
Sun introduces Java, a programming language designed to run on all operating systems.

**2007**
Microsoft releases Windows Vista, the first major overhaul of the Windows operating system since Windows XP.




The first computer program, written in 1948 by Tom Kilburn for the Mark 1, was designed to find the highest proper factor of any number. The necessary divisions were done not by long division but by repeated subtractions. It took 52 minutes to solve the problem for the number 218.

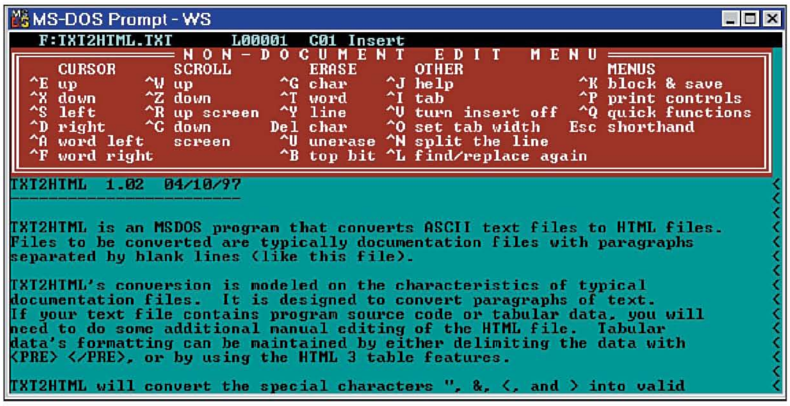
## Killer Software

When the personal computer came along, software followed a similar route. The users of the first personal computers—the Altair and the Commodore Pet—were pioneers who expected to do their own programming. They happily shared the coding tricks they learned and the programs they created. A revolution was in the making, although few realized it at the time. All the revolution needed as a spark was the "killer app."

A *killer app* is an application program so useful and popular that it can create a demand for hardware. The first killer application for the personal computer was VisiCalc. An MIT graduate, Dan Bricklen, made one of those connections that change society. He saw that millions of accountants, business managers, executives, bankers, and stock brokers were wasting their time doing something a computer can do easier and faster: number crunching. Using mechanical adding machines, people were feeding in enormous strings of numbers and crunching them into other numbers. The possibilities for error were great. And the certainty of monotonous labor was greater.

The result of Bricklen's inspiration is the electronic spreadsheet, a killer app by definition because it was easily understood—at least by people who work with numbers—and in time saved, it quickly paid for itself and a computer to run it on. On the monitor of the first truly mass-market computer, an Apple II, VisiCalc looked substantially like a paper ledger sheet. The hapless accountant still had to enter the numbers to be crunched by the computer, but he didn't have to add, subtract, divide, or multiply them. VisiCalc did that according to *formulas*—another type of program—which the computer user types into "cells," those little rectangles created by the grid of a ledger's lines. The formulas, of course, were created by humans, and there was always the possibility of human error. But once you got the formula right, you didn't have to think about it again, and you didn't have to worry about math mistakes. Just as importantly, you had to enter each relevant number only once. If a five percent sales tax was added to each purchase, the spreadsheet user had only to enter it once for VisiCalc to add the correct sales tax for each purchase.

But it wasn't only accuracy, speed, and ease that made VisiCalc so popular. It was also a matter of power, something business people understood even if they had only the vaguest idea of how these new electronic spreadsheets worked. Until VisiCalc, when a business manager wanted information about customers, sales, inventory, or budget, the manager had to send a request to the IS department, which would get around in the next few days to loading a program and data from huge spools of magnetic tape, letting it all churn for a while and then spitting out the information in the form of cards with holes in them or long streams of green and white striped paper. Wars have been fought in less time than it took most managers to get a budget projection.




The first word processing program to make it to stardom was WordStar, a DOS-based text program notorious for shortcut keys, visible codes, hidden tricks, and the fact it could easily be modified.

The personal computer changed all that. Within a few minutes, any business person with a passable knowledge of algebra and some fortitude could crunch his own numbers. Knowledge was literally at his fingertips. And in modern commerce, knowledge was valuable. It was power.

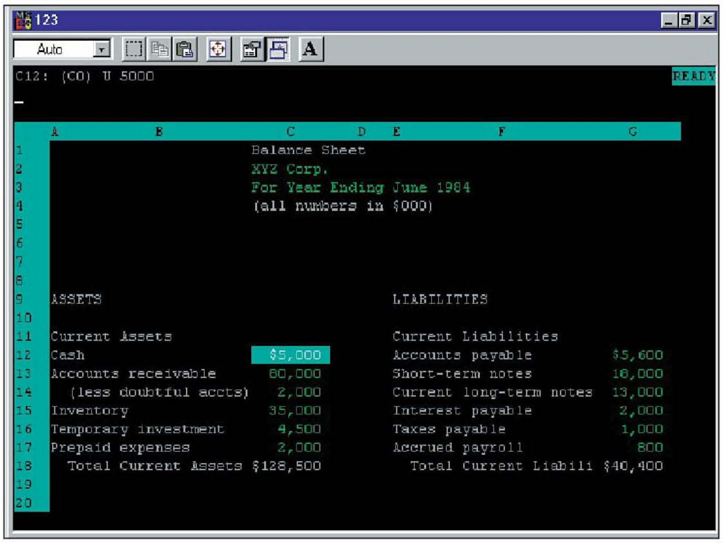
That lesson was not lost on the people managing the traditional corporate computer centers. They saw personal computers as threats to their own power bases. If anyone could juggle their own numbers, why would the company need a big, expensive department dedicated to information? The computer managers, of course, claimed that they were only trying to avert catastrophes at the hands of information amateurs. The result was that many PCs were bought with purchase orders claiming the PCs were electric typewriters.

There was another lesson in VisiCalc. The software market was to become something so volatile, so innovative, and so important that no application remained top killer for long. When the IBM PC debuted in 1981, it could not run the software then available for the Apple. Bricklen's company was slow in creating a version that would run on a PC. Before anyone knew it, a new killer app dominated the scene.

Lotus 1-2-3, introduced in 1982, was, like VisiCalc, an electronic spreadsheet. But it expanded the definition of spreadsheets by adding capabilities for graphing the data in the spreadsheet and manipulating its information the way a database manager does. The use of graphics was particularly impressive for a computer, the IBM PC, that was designed basically to display only text. Lotus 1-2-3 justified the buying of IBM PCs and compatibles in many offices. It also revealed the perils at that time of buying IBM clones. 1-2-3 took advantage of some quirks in the IBM PC to speed up performance.

But the same quirks weren't duplicated in imitation IBM PCs, with the result that 1-2-3 wouldn't work on some systems. Computer builders were forced to reproduce what they considered flaws in the IBM machine to be sure they could run any software the IBM PC could run.

The 1980s were a decade of explosive growth—and often rapid death—of computer makers and software publishers. Compared to today's machines and operating systems, the early PCs were simple enough so that one programmer working nights in a bedroom could turn out software that was killer. Software with eccentric names such as Electric Pencil and Volkswriter, both word processors, quickly expanded the capabilities of PCs. The early success of dBASE II, a database manager, inspired countless other programs with "base" in their names. Inevitably, someone created 4-5-6, which added more features to 1-2-3. Many programs filled in functions PCs were lacking. Peter Norton launched a series of products with Norton Unerase, capitalizing on the fact that the PC's operating system neglected to provide a way to retrieve a file that had been deleted accidentally.




**Lotus 1-2-3**

Lotus 1-2-3 was the first big business hit on the IBM PC because it combined the functions of an electronic spreadsheet, a database, and graphing.

***Courtesy of Lotus Corp.***

At the same time commercial software was burgeoning into every possible niche, another software movement grew alongside the programs from big corporations. In 1982, Andrew Fluegleman created PC-Talk, a program allowing computer users to communicate over phone lines. Despite the name, you did not use PC-Talk by talking. Conversations were typed, but you could also transfer entire programs from one computer to another. The decisive feature of PC-Talk, however, was the fact you could get it for free, and it established an important category of software: freeware, which later became known as shareware. Freeware programs were distributed without charge over electronic bulletin boards accessed with a modem and at the meetings of computer clubs called user groups. Often, freeware was totally free. Other shareware gave you a chance to try the program risk free, but if you liked the software and continued to use it, you were expected to send some money to its author. At $5–$40, the suggested prices were still cheap compared to software sold in boxes at stores. Some programmers humbly suggested users send whatever they thought the program was worth. Not unsurprisingly, a lot of people used the programs without paying. But enough people were honest enough and enough shareware was good enough that at least a few programmers made respectable livings. The shareware tradition continues today on the World Wide Web, although now programs you download from the Internet are likely to have some features disabled or stop working after a month to encourage users to pay registration fees.

The abundance of software in the early years also meant confusion. Many programmers had *idiosyncratic* concepts of *what* software should do and how *it* should do it. And they found equally idiosyncratic users. WordStar, for example, quickly dominated the word processor market despite a system for formatting documents based on starting lines with a period—hardly an intuitive approach. But WordStar could be modified by anyone with scant programming knowledge, a bit of daring, and a different idea of how a word processor should work.

## Power to the Software

In fact, the dominating factor in the early software leaders was not that it came out of the box as an ideal product. Instead, programs like WordStar, 1-2-3, dBASE, and WordPerfect could be manipulated by non-programmers. This was the opposite extreme from the days of big iron, when a bright idea for a way to handle information had to be submitted to the computer pros, who would get around to it Real Soon Now. The factor distinguishing software from all the helpful tools invented before it was that software was malleable. You weren't limited to using it as intended. One Lotus 1-2-3 user devised a way to use the number cruncher as a word processor. Users of WordPerfect routinely pressed the word processor's mail merge functions into duty as a database manager.

What really accounts for the popularity of the best software is the empowerment it gives people over computers. The best software came with macro or scripting capabilities that gave a layman user enormous power over this new machine. Macros let you record a series of keystrokes or write a list of simple commands for the software to follow. You made the software work the way you thought it should. You could create your own menus, fill in rows of information with a couple of keystrokes, or make your computer connect to a bulletin board service to download the latest games as you slept. There is a physical, Frankensteinian thrill to creating a macro in 1-2-3 or writing a script in CrossTalk and watching your computer obey your instructions like a faithful genie.

There was, of course, despair as well as thrill. The truly insane greatness of the 80s was as confusing as it was empowering. There was no master plan, no one person in charge of creating software. This often led to brilliant breakthroughs born of an individual spark of genius. But it also led to a new round of learning with each new program. In some programs, you might display a help screen by pressing Alt+H. Or Ctrl+H, or F7, or F1. (1-2-3's F1 came to be accepted by other programs as the standard help key because of 1-2-3's dominance and the universal need for help.)

Although some other conventions arose among programs, diversity dominated. Only with Microsoft's domination of the operating system—first with MS-DOS and today, Windows—has some semblance of order been imposed on the wild garden that software was throughout the 20th century.

# Operating Systems and Other Software

Operating systems are important enough to warrant their own chapter. But a brief explanation of what an operating system was and what it evolved into will give you a better perspective on modern software blessings and annoyances.

First of all, an operating system is fundamentally different from all other types of software in that an operating system is the one program you must have to do anything with your PC. No computers need a word processing program or a spreadsheet or the game *World of Warcraft* just to work. But each PC must have an operating system.

### OS/2

Operating System 2 (OS/2) began in 1983 as a joint project between Microsoft and IBM for high-end computers. Windows would be the operating system for entry-level PCs. Ten years later, the two companies let the joint development agreement expire. Windows and OS/2 competed for the same market, but OS/2 lost the numbers game. For the first time, a computer company other than IBM carried the standard to which other developers conformed.

On its own, a computer can do little more when you turn it on than wake up and, like an infant, search for something to feed it. The food of computers is software, and the appetizer is always the operating system. When a computer boots, it only has enough code built into it to look on a disk for a few crucial operating system files. It pulls those into memory, and they in turn load the rest of the operating system. The loaded operating system establishes rules by which the computer can then load other programs and work with hardware that the computer isn't smart enough to handle on its own.

The name for Microsoft's DOS, for example, comes from disk operating system, or OS. Originally, DOS was designed as a software tool to create disk files, copy files, delete them, and organize them. All this was done by typing text commands on a black screen. More significantly, DOS let you run other software simply by typing the name of a program. Today Windows does the same thing with a click on a tiny cartoon.

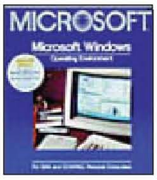
You can look at the operating system as an office's mid-level manager. The operating system posts the office rules that other programs must obey. These other programs are called applications—which roughly means anything that isn't DOS, Windows, UNIX, System 8, Linux, or the few other operating systems out there. Think of applications as being an office's top executives. Like most executives, the applications have a lot of bright ideas but not the slightest clue about how to carry them out. But applications, like executives, have someone to do the real work. Applications give the OS some general, vague instructions—such as "Save this file with the name mybudget." The operating system, in turn, passes more detailed instructions on to the office's clerks, who are the ones who really get their hands dirty. In our little metaphor, the clerks are the computer codes contained in the BIOS, dynamic link libraries, and drivers. These are the people/code who know how the office actually runs—how to record a file or add two numbers.

All applications must have an operating system or else you couldn't run them at all. But the same isn't true of operating systems. Even if Windows is the only software on your PC, you can still use the computer. This is becoming increasingly true as Microsoft integrates into Windows functions that were once performed only by separate programs, such as faxing, word processing, disk compression, and Internet browsers. The broadening of the definition of an operating system is a legal as well as a technical job. For better or worse, it's increasingly true that although applications must have an operating system to run, an operating system doesn't need application programs to be useful.

# The Devolution of the OS

If you've been using a PC long enough to know what a DOS prompt is, then you know that MS-DOS wasn't designed to be easy to use. Actually, DOS was only one of two operating systems for the PC. Initially, IBM sold the PC with both Microsoft's DOS and CP/M-86, an upgrade for the 8086 processor of CP/M (Control Program for Microcomputers). CP/M was a down-and-dirty operating system found on several brands of microcomputers at the time. Both DOS and CPM-86 were more concerned with providing a learning path that would be familiar to people who already knew CP/M, the original computer geeks.

Microsoft and other companies realized early on that this shortcoming was a roadblock to more widespread user acceptance, and there were continual attempts on several fronts to improve or replace DOS with an operating system that extends the computer's capabilities and make it easier to use. Some operating systems, such as GEOS and DR DOS, gained cult followings but never had the market commitment needed to overtake Microsoft's own evolving operating system. IBM and Microsoft split over their joint development of a new graphic operating system, OS/2. Following the split, IBM continued to develop and sell OS/2, but even IBM's clout was no match for Microsoft's marketplace entrenchment.




**Windows 1**

In 1985, Microsoft introduced Windows, a graphic interface that let you make selections with a mouse from menus and run more than one program at a time. It was slow, required major memory, and received a universal yawn.

What an operating system needed more than bells and whistles—what it had to have—was the capability to run the thousands of programs that had already been developed under DOS. Software developers were loath to jump any bandwagon unless they were sure the public would follow. Microsoft had the only working bandwagon in town. When Windows first appeared in November 1985, it was slow, crash-prone, and demanded processing and memory resources that were then prohibitively expensive. It used graphics and a mouse pointer to do the jobs that required obscure typed commands in DOS. It was an unimaginative copycat of the Macintosh operating system. Still, Microsoft, better than any competitor, could guarantee that billions of dollars of software and business data already created under DOS would still be usable.

Technically, Windows was at first an "operating environment," window-dressing that hid DOS continuing to do the dirty work behind the curtains. And the dirtiest of them all was running program code that was originally designed for the PC's first processor, which could at best manipulate only 16 bits of data at a time. Many programs written for DOS depended on that 16-bit mechanism even after Intel introduced the 80386, which could process 32 bits of data at once. DOS and even more recent Windows variants like Windows 95, 98 and Me, had to maintain parts of its own code that handled 16-bit operations. Microsoft did develop a variant of Windows—Windows NT, which became Windows 2000—that ran only 32-bit instructions. NT was a more stable variation of Windows, but it couldn't run older applications designed for older processors and it is found mostly on *file servers*, brawny computers that feed files to hundreds of other computers on a network.

Despite the deservedly cold reception Windows first received, Microsoft kept revising Windows until, with version 3.1 in June 1992, it had an operating system that, although still far from ideal and still tied down by 16-bit code, was fast enough and stable enough to convince the public and software publishers to hop on.

The biggest step for Windows came in 2001, when Microsoft updated its consumer-oriented OS to Windows XP, a fully 32-bit operating system, giving it the speed, features, and security of Windows 2000. If Windows XP was about speed and power (the improved security was still lacking), then five years later Windows Vista's appearance was largely about…appearance. With see-through windows and dialog boxes decorated with sweeping fields of color and endowed with endlessly cute gadgets to give toys to all levels of computer users, there was very little even the most die-hard disparager of Windows could find to complain about. Even security issues got a potent and needed boost.

Today, Windows has no serious challenge as the mainstream operating system, although Microsoft, beset with charges of monopolism, has perfected the art of trembling and pointing fingers at Java and Linux. Java is a programming language that is popular on the Internet, and is accessible by computers running every operating system imaginable. Linux is a version of Unix, a geeky operating system for high-powered computers. Linux has gained popularity more on philosophical than practical grounds. It is free, and the underlying code is available to anyone. Other programmers are encouraged to make improvements to the code and share them with other Linux users. Linux has the idealism found in the earliest programmers who considered selling software for profit the equivalent of crossing over to the Dark Side. It's even gaining a market share in use on file servers. But despite generic ports like Linux, the future of the operating system is devolution. Instead of evolving into a multitude of varied species, each riding its own varied strengths, the few primitive operating systems of an earlier era have devolved into even fewer species. The most serious threat to Windows might come from an unlikely source: the Internet. You can already see the distinctions starting to fade between what happens on the desktop and what happens on the Internet. Sun and Corel are providing software that operates partly in one arena, and partly in another. Eventually, the concept of an operating system could dissolve into tiny pieces of software residing on our hard drives and on the Internet that are called upon to furnish various services by other pieces of software or by computer hardware. The operating system might simply become part of the background—an important part, like highways are to transportation, but something we rarely think about except when there are traffic jams and holes.

In this part of the book, we'll look at three types of software. First, the programming languages that software developers use to create other software. You might never use programming languages, but understanding how a program is created can help explain some of the inexplicable eccentricities of your computer. Then we'll look at Windows and how the operating system functions as an ambassador between hardware and other software and as a straw boss riding herd on the third type of software we'll look at—applications.

## KEY CONCEPTS

**AI (Artificial Intelligence)** AI is used in games for everything from making a computer opponent behave believably like a human opponent to having automated units perform tasks in a realistic manner.

**algorithm** A procedure, rule, or formula for solving a problem. A computer program is essentially an elaborate algorithm that goes through a series of steps to arrive at a specific outcome.

**application** Software that performs a specific function that is the end result of using the computer. Word processors, database programs, Web browsers, and image-editing programs are all applications.

**beta version, or beta** A working version of a program being developed in which most of the components work. A company often sends betas to its customers, who try to find bugs in the beta. A program might go through several betas before being released to the public.

**bug** A flaw in software that causes the program to malfunction or freeze when it encounters specific situations.

**call** A programming command that uses—calls on—a specific routine elsewhere in the program or in a .DLL file. After the routine has ended, execution returns to the point in the code where the call happened.

**compiler** A software tool that converts source code into a format that can run only with the assistance of an operating system. Contrast with **interpreter**.

**dynamic link library (DLL)** Collections of code all housed in the same DLL file that can be used by more than one program.

**dialog box** An onscreen display that allows a computer user to select among several choices to determine how a program should operate.

**flow** The sequence of functions a program follows.

**flow chart** A diagram that depicts the flow of a program.

**game engine** A computer game's master program that coordinates all the other components, such as graphics, sound, calculating, and physics.

**graphic user interface (GUI)** A method of controlling software using onscreen icons, menus, dialog boxes, and objects that can be moved or resized, usually with a pointing device such as a mouse.

**high-level language** Software code written with recognizable words that more closely resemble convention languages, such as English. See **low-level language**.

**interface** The design of a program that determines how a computer user interacts with software. The two most common interfaces are the graphic user interface and the text interface, which consists of typing command words.

**interpreter** A software tool that converts source code on the fly into instructions the computer can understand. Contrast with **compiler**.

**loop** A section of software commands that returns to its beginning and runs again as long as no specific condition has been encountered.

**low-level language** Software code written with specialized words that bear little resemblance to ordinary language but which require less interpretation or compilation to create a finished program.

**module** A generic term for self-contained sections of a program that perform specialized functions, such as a spelling checker.

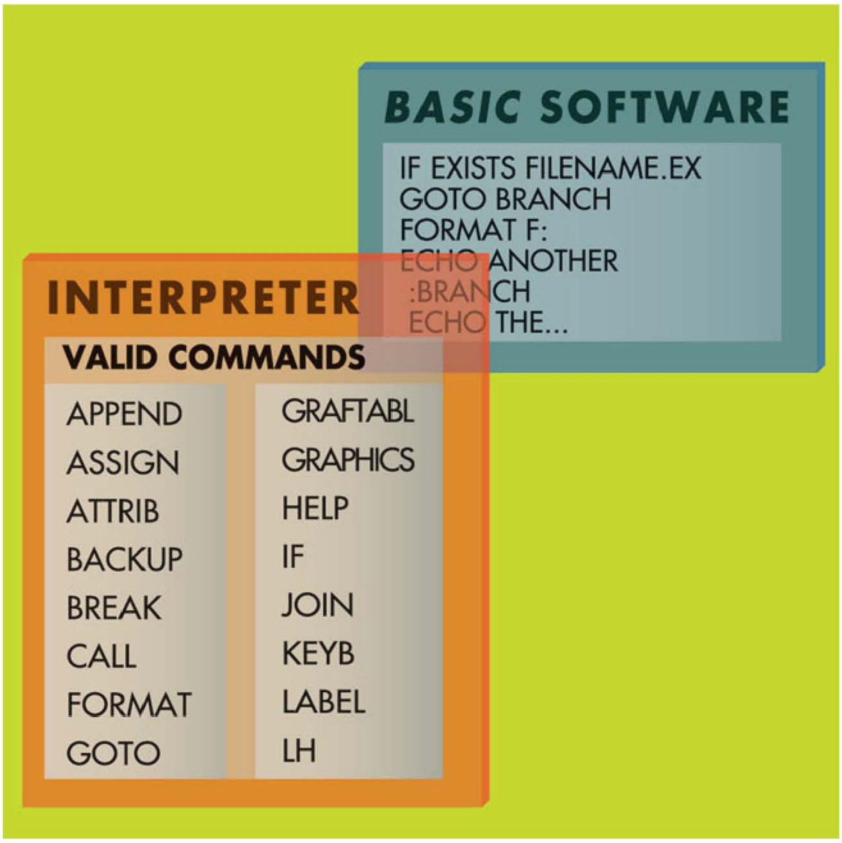
**multithreaded** A type of program designed to take advantage of Intel processors' ability to execute more than one string of data—thread—at the same time. Multithreaded programs must be designed so the threads can run independently and will not interfere with each other.

**operating system** The software that serves as a bridge between the computer hardware and application software with which the computer user works.

**runtime** The time a program or routine is being executed, or running. A program that is not dependent on an interpreter or other program the computer user works with.

CHAPTER 7

# How Programming Languages Work



**OUR** most fundamental tool as intelligent beings is language. It is through language that you learn new information and share your knowledge, feelings, and experiences with others. Through language, you can express any thought anyone has ever had and describe any event, real or fictional. The world is controlled through language. Presidents to petty functionaries, generals to GIs, CEOs to clerks—all rely on language to give instructions to others and to gather information.

Language is a necessity for a computer, too. Software is created using special languages that provide instructions for telling the computer what to do. And language defines the data with which the instructions will work. Computer language is similar to human language in many ways. The nouns, verbs, prepositions, and objects found in English, for example, have their counterparts in programming code—or the source code, the actual lines of text that get translated into functioning programs. But software sentences have their own syntax, and the words that make up the languages have their own precise meanings.

Computer language is more exacting and more limited than English. An often-repeated story tells how, in an early attempt to use a computer to translate English into Russian, the phrase "The spirit is willing, but the flesh is weak" was interpreted as "The vodka is ready, but the meat is rotten." The story might be mythical, but it illustrates a reality—that computers and their languages do not do a good job of managing the ambiguities and shades of meaning in human language that any four-year-old understands (although advances in voice recognition have computers understanding what we say, if not what we mean).

If programming languages lack the subtleties of human language, human language cannot match the precision of computer-speak. Try, for example, to describe a simple spiral without using your hands. It's impossible in English. But because math is an integral part of computer languages, those languages cannot only describe a spiral but also can provide the instructions to create an image of that spiral on a display or printer.

## Different Programming Languages

Just as there is more than one language for humans, so is there more than one computer language, even for the same type of computer. Generally, the various languages are described as low-level or high-level. The more a computer language resembles ordinary English, the higher its level. Lower-level languages are more difficult to work with, but they usually produce programs that are smaller and faster.

On the lowest level is **machine language**. This is a series of codes, represented by numbers (ones and zeros), used to communicate directly with the internal instructions of the PC's microprocessor. Deciphering machine language code or writing it is as complex a task as one can tackle in computing. Luckily, we don't have to do it. Programs called **interpreters** and **compilers** translate commands written in higher-level languages into machine language. We'll look at both interpreters and compilers later in this chapter.

On a slightly higher level than machine language is **assembly language**, or simply **assembly**, which uses simple command words to supply step-by-step instructions for the processor to carry out. Assembly language directly manipulates the values contained in those memory scratch pads in the microprocessor called **registers**. In machine language, the hexadecimal code 40 increases by one the value contained in the register named AX; assembly language uses the command INC AX to perform the same function. Although assembly language is more intelligible to

humans than machine language codes, assembly is still more difficult to use than higher-level languages. Assembly remains popular among programmers, however, because it creates compact, fast code.

On the high end, languages such as C and Java allow programmers to write in words and terms that more closely parallel English. And the programmer using these languages need not be concerned with such minutiae as registers. The C language is powerful and yet reasonably simple to write and understand. Currently, Java is the rising star among languages because a program written in Java will run on any computer no matter what its operating system. This is a distinct advantage when you're writing programs people will use over the Internet, using anything from PCs to Macs and Sun workstations. Software written in C, in contrast, must be modified to allow a program written for one type of computer to be used on another.

At the highest level are languages such as BASIC (Beginners All-purpose Symbolic Instruction Code), Visual Basic, the DOS batch language, and the macro languages used to automate applications such as Microsoft Office and Corel WordPerfect Office.

## Software Construction

A program can be a single file—a record of data or program code saved to a disk drive. But generally, complex software consists of one file that contains a master program—the **kernel**—surrounded by a collection of files that contain subprograms, or **routines**. The kernel **calls** the routines it needs to perform some task, such as display a dialog box or open a file. A routine can also call other routines in the same file, in another file that's part of the program, or in files provided by Windows for common functions. Together, the kernel and subprograms give programs a way to receive, or **input**, information from the keyboard, memory, ports, and files, rules for handling that input data, and a way to send, or **output**, information to the screen, memory, ports, and files.

Typically, when a user types information into a program, it is stored as a **variable**. As the term suggests, the information a variable stores varies from one instance to another. Programs on their own are also capable of storing in variables the information based on the results of a calculation or manipulation of data. For example, to assign the value 3 to a variable X, BASIC uses the command X = 3. Assembly language accomplishes the same thing by assigning the value to the AX register with the command MOV AX,3. Some languages require several commands to achieve the same effect another language accomplishes with a single command.

After a program has information in a variable, it can manipulate it with commands that perform mathematical operations on numbers or parse text strings. **Parsing** is the joining, deletion, or extraction of some of the text characters to use them elsewhere in the program. When a variable is text, it is often called a **string**. You can have math strings, but most often, the term *string* refers to an uninterrupted series of alphanumeric and punctuation characters. Through parsing, a program can locate, for example, the spaces in the name "Phineas T. Fogg"; determine which parts of the string make up the first name, the middle initial, and the last name; and assign each segment to a separate variable. A typical math manipulation would be X = 2 + 2, which results in the variable X having the

value 4. If that command is then followed by X = X + 1, the new value of X would be 5. The command X = "New" + " " + "York" assigns the string "New York" to variable X.

Programs can rely on the BIOS (see Chapter 3) to perform many of the input and output functions—such as recognizing keystrokes, displaying keystrokes onscreen, sending data through the parallel and serial ports, reading and writing to RAM, and reading and writing disk files. The programming language still must have commands to trigger the BIOS services. Consider the following series of BASIC commands:

```
OPEN "FOO" FOR OUTPUT AS #1
    WRITE #1, "This is some text."
    CLOSE #1
```

These commands create a file named FOO that contains the text in quotes: This is some text. It then closes the file. The language Pascal does the same with these commands:

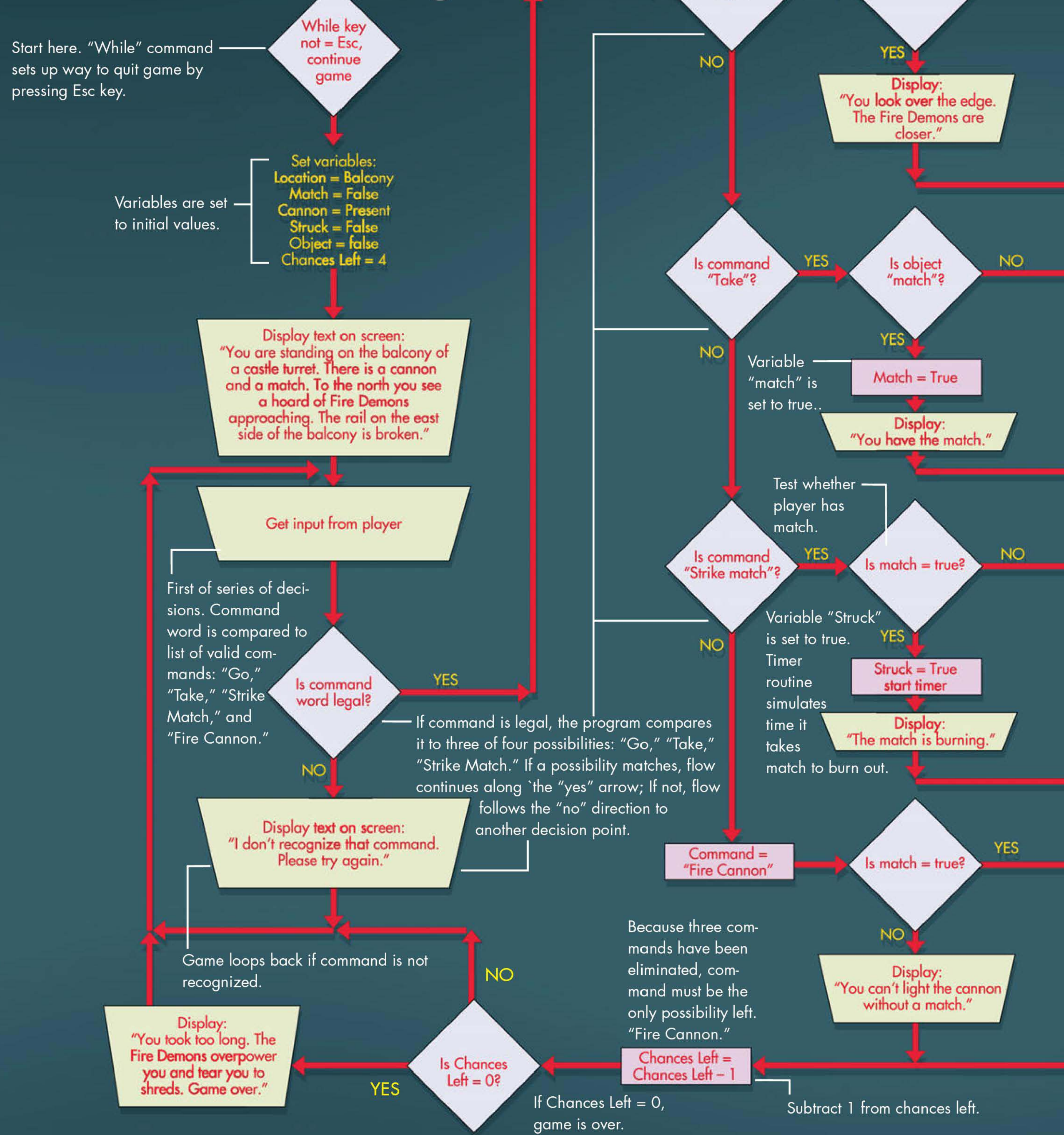
```
Assign (TextVariable, "FOO");
    WriteLn (TextVariable, "This is some text.");
    Close (TextVariable);
```

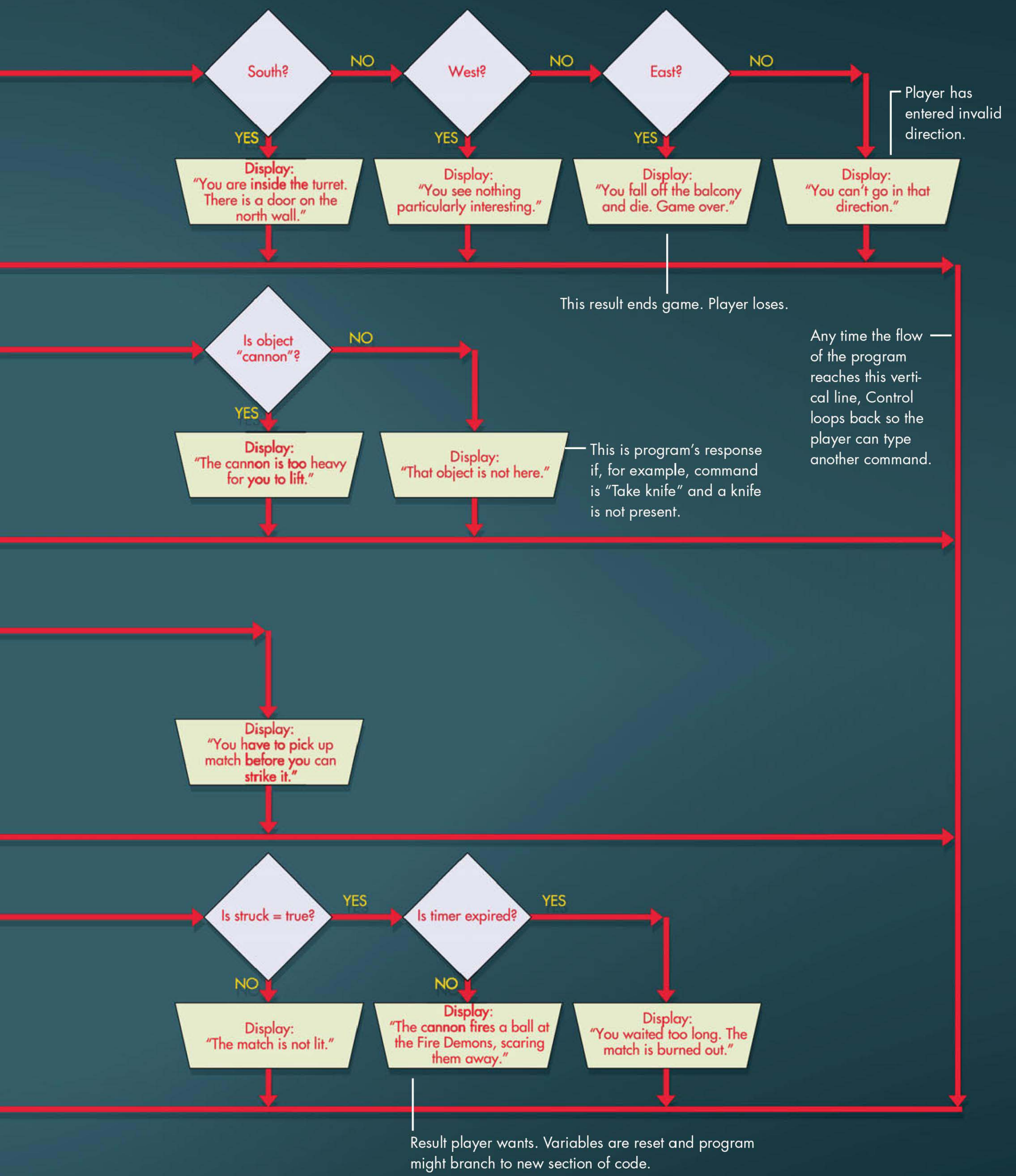
So far, we've described a fairly straightforward scheme: in, process, out. The reality is more complex. A program must be capable of performing different tasks under different circumstances—a feature that accounts for programming languages' power and versatility. And because a program hardly ever proceeds in a straight line from start to finish, there are commands that tell the computer to **branch** to different parts of the program to execute other commands. In BASIC, the command GOTO causes the execution to move to another part of the program. Assembly language does the same with the command JMP (short for jump).

Branching is used in combination with the **Boolean logic** functions of the programming languages. For example, when a program needs to change what it's doing because a particular condition exists, it can use "**if...then...**" The program checks to see whether a certain condition is true, and if it is, the program then performs a certain command. For example, if the variable State is the string "Texas," the program uses the abbreviation TX to address a letter.

To get an idea of how programs are written, we'll look at a **flowchart**—a kind of map sometimes used by programmers to lay out the logical connections among different sections of the program code. As an example we'll use a type of game that was popular in the days when text-only adventures like *Zork* were plentiful. In such games, the user types in elementary commands, such as "Go east," "Go north," "Take knife," and "Hit monster." And the game displays a sentence describing the consequences of the player's action. Our example is oversimplified and takes into account only one small portion of such a game. As such, it gives you an idea of how many commands and how much programming logic go into even the simplest code. In our adventure, the player is already on the balcony of a castle turret surrounded by Fire Demons…

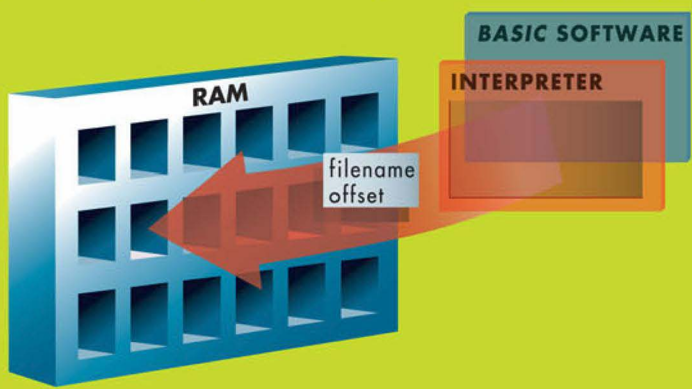
# The Flow of a Program



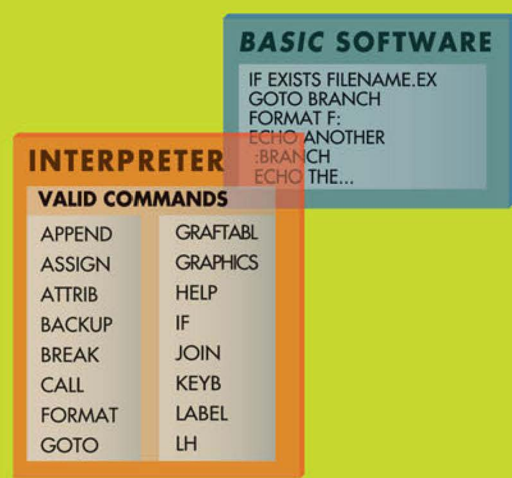

South?
NO
West?
NO
East?
NO
YES
YES
YES
Display: "You are inside the turret. There is a door on the north wall."
Display: "You see nothing particularly interesting."
Display: "You fall off the balcony and die. Game over."
Display: "You can't go in that direction."
Player has entered invalid direction.
This result ends game. Player loses.
Is object "cannon"?
NO
YES
Display: "The cannon is too heavy for you to lift."
Display: "That object is not here."
This is program's response if, for example, command is "Take knife" and a knife is not present.
Any time the flow of the program reaches this vertical line, Control loops back so the player can type another command.
Display: "You have to pick up match before you can strike it."
Is struck = true?
YES
Is timer expired?
YES
NO
NO
Display: "The match is not lit."
Display: "The cannon fires a ball at the Fire Demons, scaring them away."
Display: "You waited too long. The match is burned out."
Result player wants. Variables are reset and program might branch to new section of code.

# How Software Interpreters Work

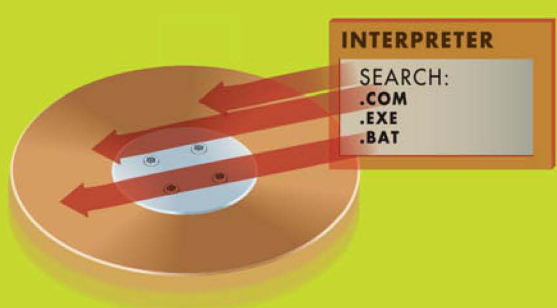
**1** Some software programs can't communicate directly with a computer. They are written in **languages** that require an **interpreter**, another program that translates the software's commands into instructions the processor can use. Interpreted programs include DOS batch files, programs written in BASIC, WordPerfect macros, and Java software written for use on the Internet. Each command in the program is written on a separate line. When you launch an interpreted program, an interpreter designed particularly for that language establishes a small area in memory. There, the interpreter puts the name of the file and keeps track of its current place, called the **offset**, which is measured in the number of lines the current command is from the beginning of the file.



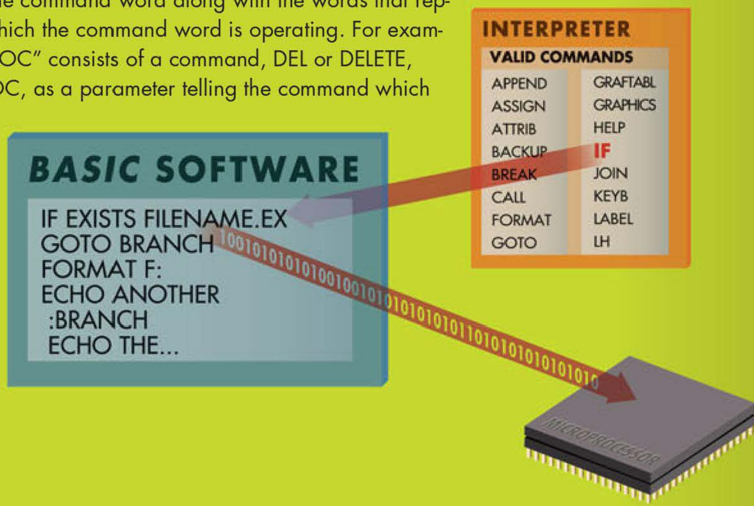

**2** As the interpreter reads each line of the file, it compares the first word in the line to a list of valid **commands**. In some instances in a BASIC program, the interpreter will also recognize a **variable** at the start of a line. A variable holds some temporary data, such as a filename or a number.




**3** In a batch file, if the first word of a line is not found on the approved list, the interpreter will look for a .COM, .EXE, or .BAT file with a name matching the word. If none of these conditions are fulfilled in a batch file, or, if in a BASIC program a matching command word or variable is not found, the interpreter generates an **error message**.

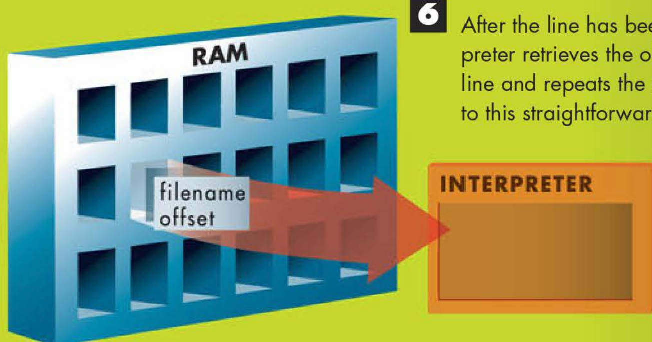


**4** If the word is found on the list of valid commands, the interpreter **executes** the entire line, translating the command word along with the words that represent the parameters on which the command word is operating. For example, the line "DEL MYFILE.DOC" consists of a command, DEL or DELETE, and a filename, MYFILE.DOC, as a parameter telling the command which file to delete. The command and parameters are turned into code **tokens—**shorthand abbreviations for instructions that are passed to the microprocessor, which carries out the instructions.




**5** If any of the parameters are invalid, or if they attempt to perform a forbidden operation, such as copying a file over itself, the interpreter generates a **syntax error** message.


**6** After the line has been processed, the interpreter retrieves the offset location of the next line and repeats the procedure. The exception to this straightforward progression occurs if a command, such as GOTO, **branches** execution to another section of the program.



# How a Compiler Creates Software

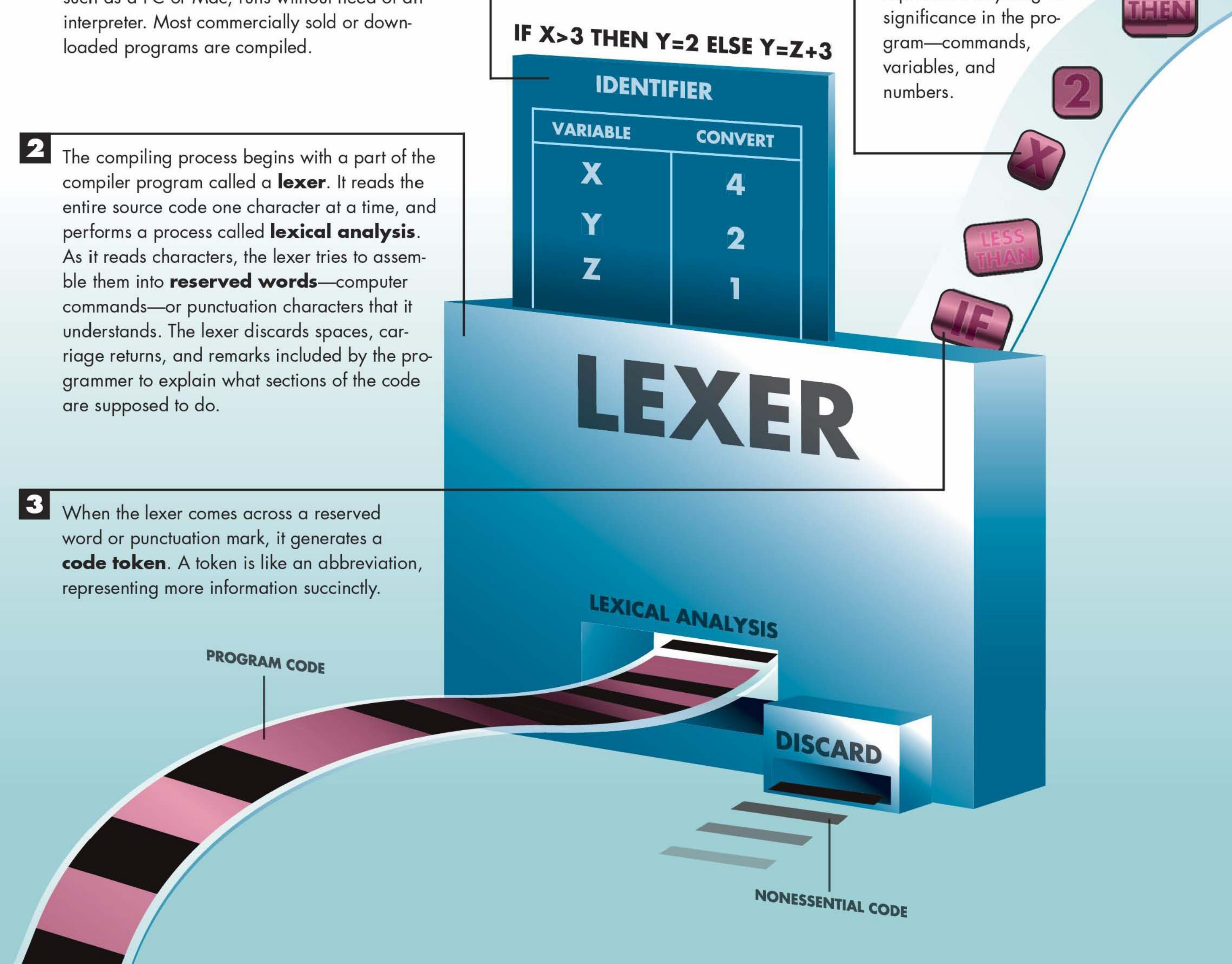
**1** The **interpreter** in the previous illustration and a **compiler** are both software programs that translate program source code that humans understand, such as BASIC or C+, into machine language, which computers understand. The difference between a compiler and an interpreter is this: An interpreter translates the source code, line by line, each time the source program is run; a compiler translates the entire source code into an **executable** file that a specific type of computer, such as a PC or Mac, runs without need of an interpreter. Most commercially sold or downloaded programs are compiled.

**2** The compiling process begins with a part of the compiler program called a **lexer**. It reads the entire source code one character at a time, and performs a process called **lexical analysis**. As it reads characters, the lexer tries to assemble them into **reserved words**—computer commands—or punctuation characters that it understands. The lexer discards spaces, carriage returns, and remarks included by the programmer to explain what sections of the code are supposed to do.

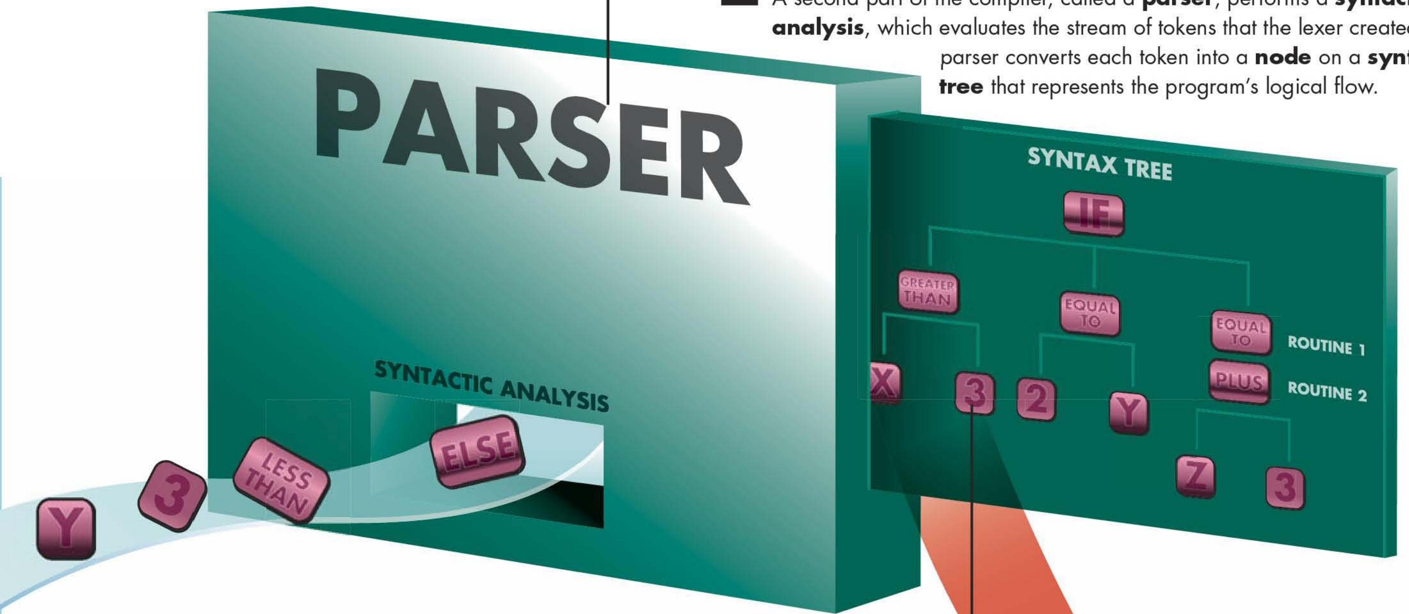
**3** When the lexer comes across a reserved word or punctuation mark, it generates a **code token**. A token is like an abbreviation, representing more information succinctly.

**4** When the lexer finds a string of characters that don't form a reserved word, the lexer assumes that those characters stand for a **variable**. It assigns the variable a place in an **identifier table** that tracks the name and contents of every variable in the program. Then the lexer generates a **variable token** that points to the variable's position in the identifier table. When the lexer finds a string of numeric characters, the lexer converts the string into an integer and produces an **integer token** to stand for it.

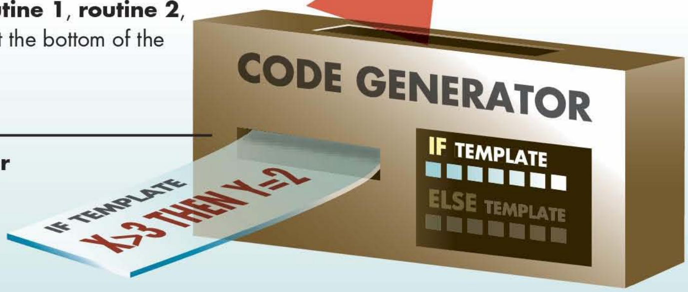
**5** The result of the lexical analysis is a stream of tokens that represent everything of significance in the program—commands, variables, and numbers.



**6** A second part of the compiler, called a **parser**, performs a **syntactic analysis**, which evaluates the stream of tokens that the lexer created. The parser converts each token into a **node** on a **syntax tree** that represents the program's logical flow.

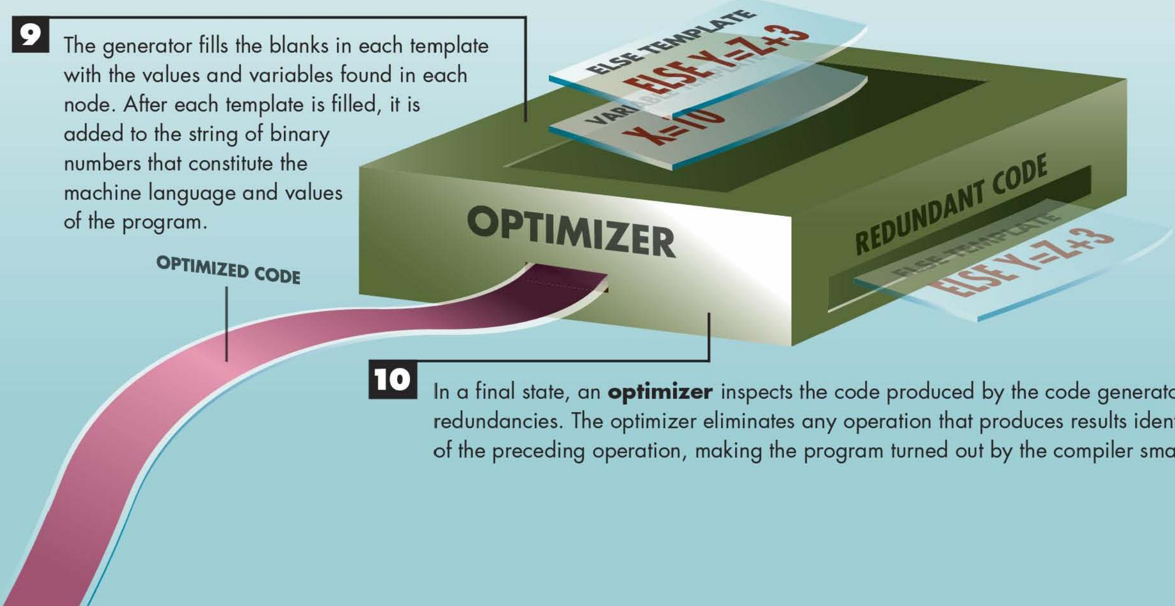



**7** Each node on the tree represents a program operation that generates data or an instruction that is passed to the node above it. The node, in turn, performs another operation and passes that result to the node above it. When the parser is finished, the compiler has converted the entire program into a tree that represents the program's structure. The topmost node is called the **program**, and the nodes that pass results to it are **routine 1**, **routine 2**, and so forth, all the way down to very specific nodes at the bottom of the tree.




**8** A third part of the compiler called the **code generator** works its way through the syntax tree, producing segments of machine code of each node. To each node on the tree, the generator matches a template of machine code to the operation assigned to that node.

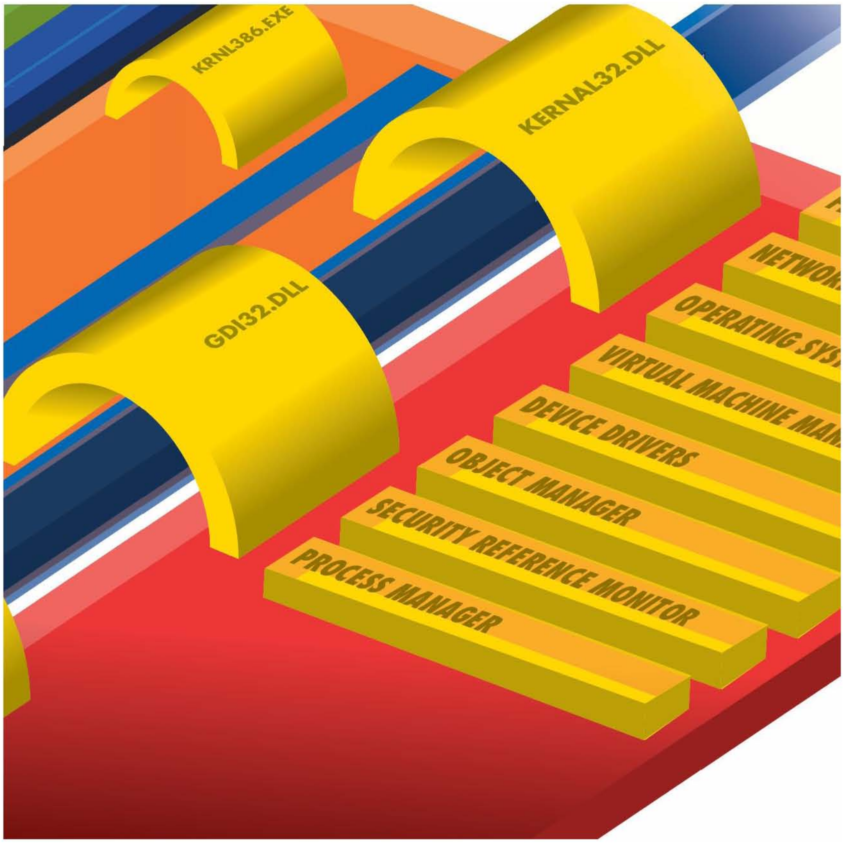
**9** The generator fills the blanks in each template with the values and variables found in each node. After each template is filled, it is added to the string of binary numbers that constitute the machine language and values of the program.




**10** In a final state, an **optimizer** inspects the code produced by the code generator, looking for redundancies. The optimizer eliminates any operation that produces results identical to those of the preceding operation, making the program turned out by the compiler smaller and faster.

CHAPTER

# 8 How Windows Works



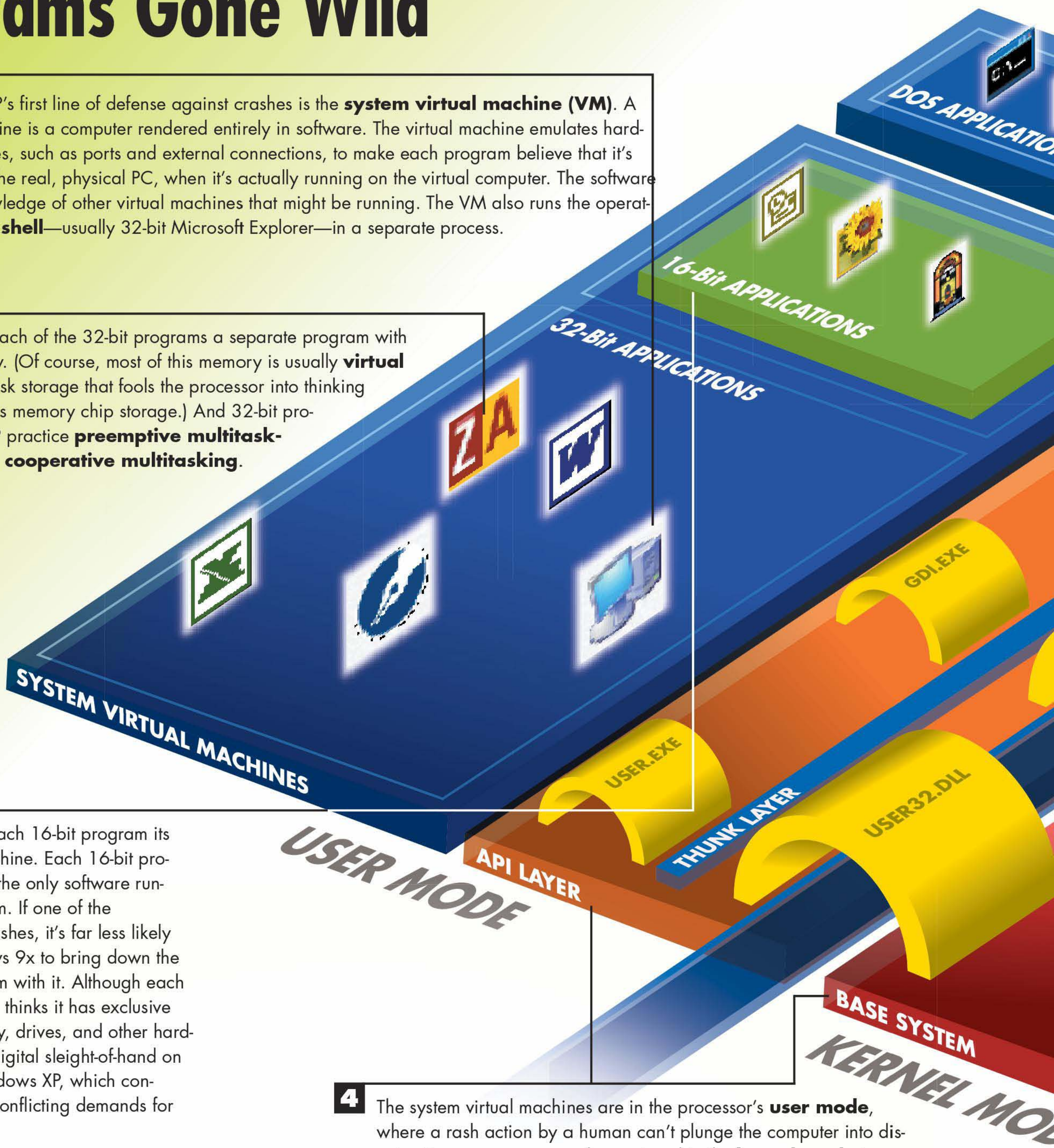
THE Windows operating system is more than just a pretty face. Sure, it's cute, with all the tiny pictures called icons, sound effects, your personal color scheme, and the capability to drag things around like pull-toys.

But behind that face is a stern taskmaster. Both Windows XP and Windows Vista, Microsoft's two most advanced versions of Windows, are not your father's operating systems. Earlier versions of the consumer level of Windows were built on top of DOS, an elementary operating system for small computers and modest programs that came with the first PCs. DOS was inept at graphics, color, and sound, features that are commonplace in Windows.

More importantly, DOS was intended to run only one program at a time. There was no competition by different programs for the attention of the processor, memory, and drive storage. Computers back then might not have had much memory, but any program loaded into that memory had the run of the place. Now, these programs were expected to play nice and share such toys as RAM and the CPU. They didn't, of course. And when programs trampled on another program's RAM, Windows didn't know what to do and collapsed into the dreaded Blue Screen of Death.

With XP and Vista, Windows finally got smart—and sneaky. As you'll discover in this chapter, Windows became enormously more stable when it used pretend computers so that each program thought it was the only program running on the PC. You'll also see how Windows is more than one program. It's a complex, intertwined *organization* of scores of programs, more like a hive than a single worker.

# How Windows XP Controls Programs Gone Wild

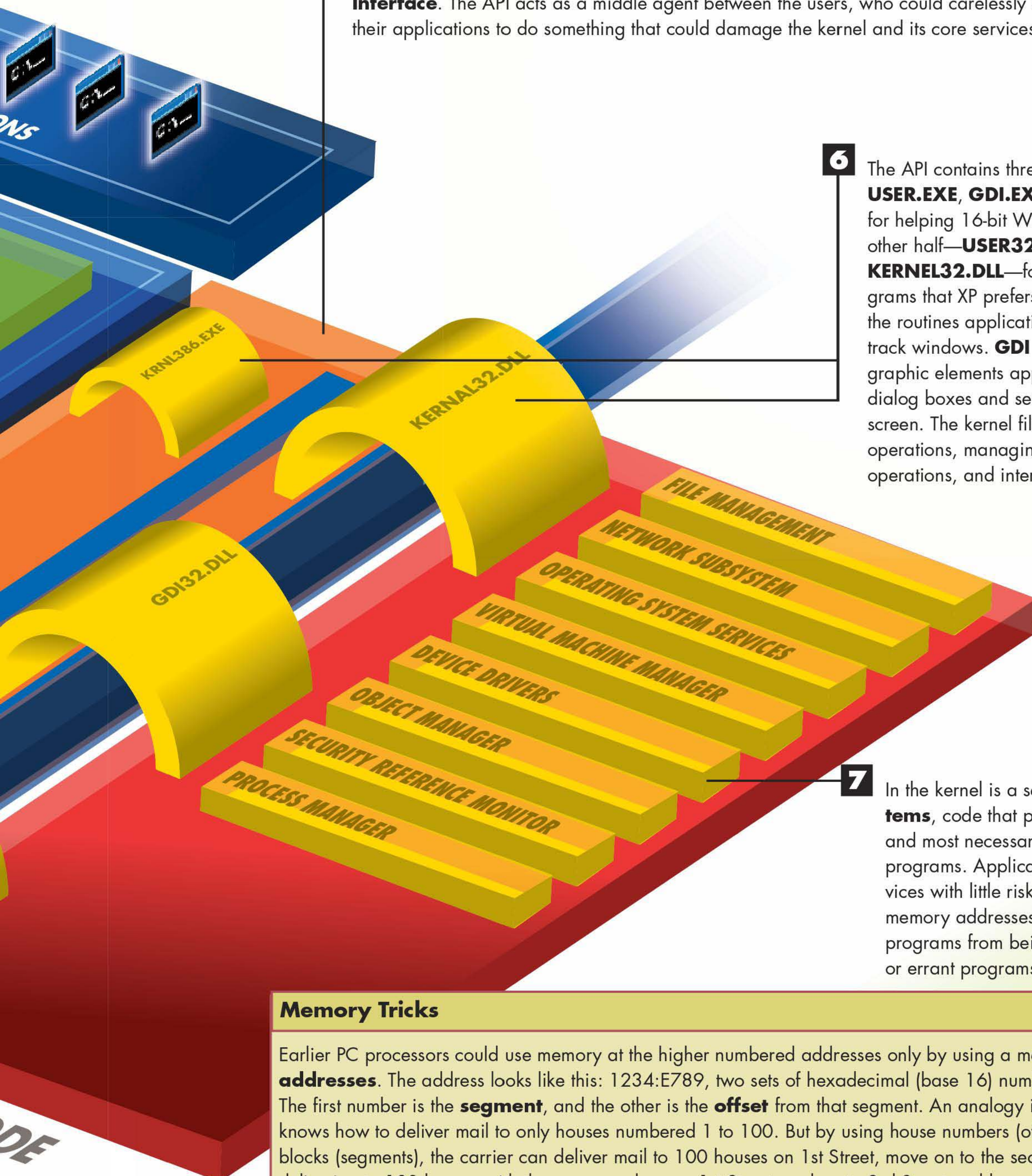



**1** Windows XP's first line of defense against crashes is the **system virtual machine (VM)**. A virtual machine is a computer rendered entirely in software. The virtual machine emulates hardware features, such as ports and external connections, to make each program believe that it's running on the real, physical PC, when it's actually running on the virtual computer. The software has no knowledge of other virtual machines that might be running. The VM also runs the operating system's **shell**—usually 32-bit Microsoft Explorer—in a separate process.

**2** The VM gives each of the 32-bit programs a separate program with 4GB of memory. (Of course, most of this memory is usually **virtual RAM**—drive disk storage that fools the processor into thinking the disk space is memory chip storage.) And 32-bit programs under XP practice **preemptive multitasking** rather than **cooperative multitasking**.

**3** The VM gives each 16-bit program its own virtual machine. Each 16-bit program thinks it's the only software running on a system. If one of the applications crashes, it's far less likely than in Windows 9x to bring down the rest of the system with it. Although each of the programs thinks it has exclusive rights to memory, drives, and other hardware, it's only digital sleight-of-hand on the part of Windows XP, which constantly juggles conflicting demands for resources.

**4** The system virtual machines are in the processor's **user mode**, where a rash action by a human can't plunge the computer into disaster. Dangerous commands are stored in the **kernel mode** in a rich set of instructions called the **kernel**, or **base system**. The kernel mode is off-limits to applications. To call on any 16-bit or 32-bit operations, applications must send requests through the API.




**5** The applications in their virtual machines are a rowdy bunch, digitally speaking. When any of the applications wants to call on any of Windows XP's core services, such as writing a file, the application must first go through the **API**—the **Application Programming Interface**. The API acts as a middle agent between the users, who could carelessly ask their applications to do something that could damage the kernel and its core services.

**6** The API contains three paired programs, half—**USER.EXE**, **GDI.EXE**, and **KRNL386.EXE**—for helping 16-bit Windows programs, and the other half—**USER32.DLL, GDI32.DLL**, and **KERNEL32.DLL**—for the full-blown 32-bit programs that XP prefers. The **USER** files contain the routines applications need to control and track windows. **GDI** files are collections of graphic elements applications use to build their dialog boxes and send information to the screen. The kernel files work with low-level operations, managing memory, input/output operations, and interrupts.

**7** In the kernel is a set of **services**, or **subsystems**, code that powers the most common and most necessary functions in all Windows programs. Applications can use these services with little risk of treading on forbidden memory addresses. To see how Vista protects programs from being trampled on by viruses or errant programs, see p. **116**.

### Memory Tricks

Earlier PC processors could use memory at the higher numbered addresses only by using a method called **segmented addresses**. The address looks like this: 1234:E789, two sets of hexadecimal (base 16) numbers with four digits each. The first number is the **segment**, and the other is the **offset** from that segment. An analogy is a postal carrier who knows how to deliver mail to only houses numbered 1 to 100. But by using house numbers (offsets) combined with blocks (segments), the carrier can deliver mail to 100 houses on 1st Street, move on to the second segment, 2nd Street, delivering to 100 houses with the same numbers as 1st Street, and on to 3rd Street and beyond. Reading and writing memory was slower because the processor had to take an extra step to work with the segment and the offset.

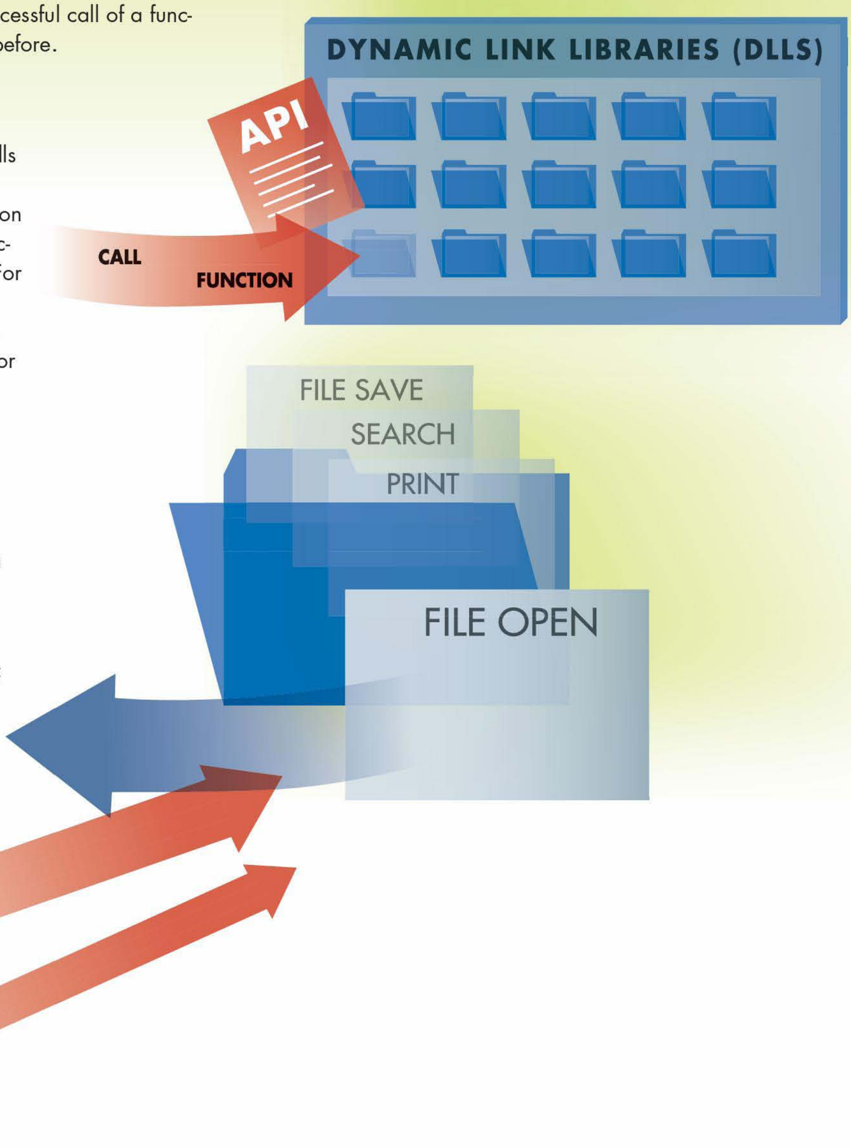
# How Windows Shares Program Code

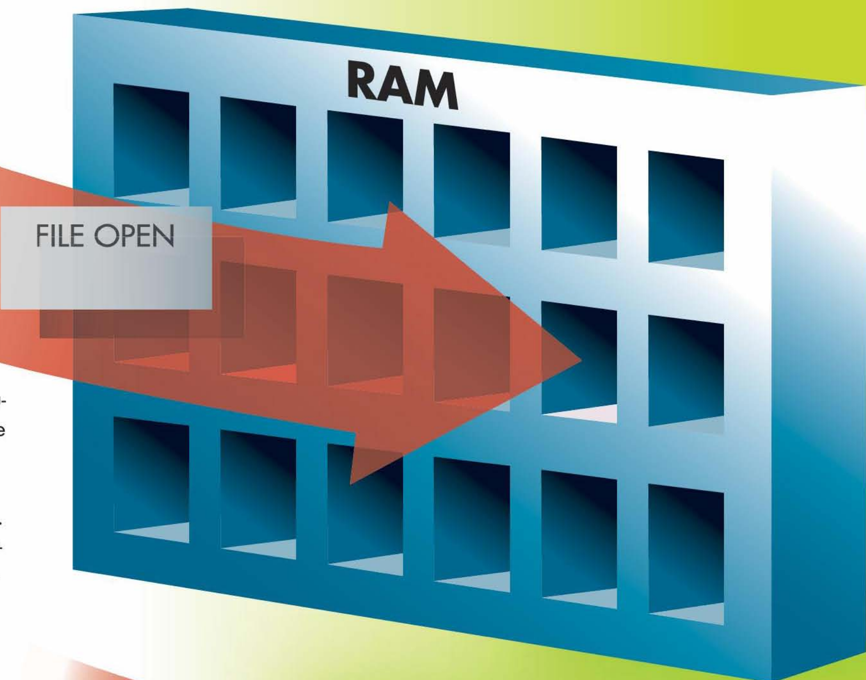
1 Windows 9x and Windows XP both provide several files called **DLLs (dynamic link libraries)**. These are collections of software code that perform common software functions. Windows has one of the most frequently used DLLs, COMMDLG.DLL. As the name suggests, the DLL specializes in commonly used dialog boxes. Its functions include displaying File Open, File Save, Search, and Print dialog boxes. Microsoft is not the only creator of DLLs for Windows. Other companies have created DLLs to provide functions such as file compression or added printing abilities.

2 An application that wants to take advantage of a DLL function first checks with an **API (application programming interface)** to find out how to call the function. All DLLs have APIs to help an application make a successful call of a function from a DLL it's never encountered before.

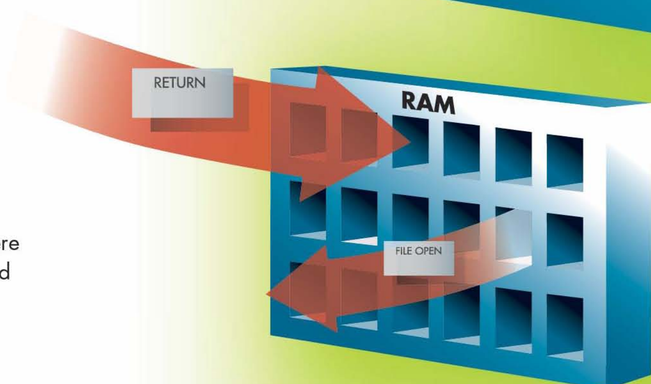
3 With the API's help, the application calls the function to life by sending it the proper digital command. The application also sends any information the DLL function needs to complete the operation. For example, a program calling the Open File function in COMMDLG.DLL passes along a file specification, such as *.* or *.DOC, to be displayed in the dialog box's Filename text box.

4 The application also passes along a specification for the type of information it expects the DLL to return to the application when the DLL has done its job. The application, for example, might expect return information in the form of integers, true/false values, or text.





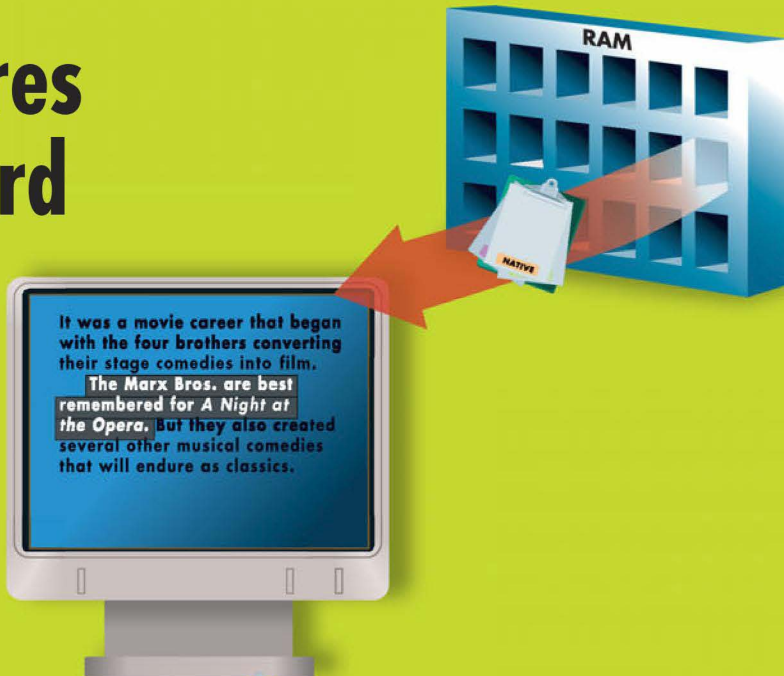

**5** The DLL loads the specific routine into memory and then the processor executes it. At this point the DLL, rather than the application, is running things. The DLL performs all the operations necessary to communicate with Windows and, through Windows, with the PC's hardware.




**6** After the DLL function is complete, the DLL puts the return information into memory, where the application can find it, and instructs Windows to remove the DLL routine from memory.

**7** The application inspects the return information, which usually tells whether the DLL function was able to execute correctly. If the operation was a success, the application continues from where it left off before issuing the function call. If the operation failed, the application displays an error message.

# How Windows Shares Data on the Clipboard

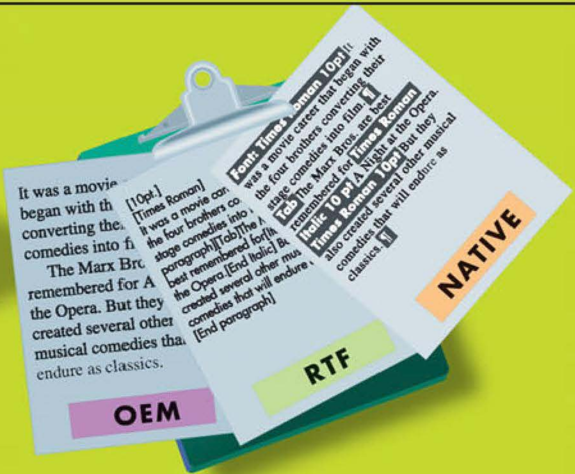



**1** The simplest way to share the same data among different documents and different applications is through the Windows Clipboard. Any time you select some data—text, graphics, spreadsheet cells—and copy it, Windows places a replica of that data into a section of memory reserved for its Clipboard. Actually, Windows creates three different versions of the data.



**2** Applications save their documents in different **formats**—the exact coding that defines how the data is structured. When you copy or cut data, Windows transfers the selection to the Clipboard in multiple formats so that it can then be pasted into applications that use different formats. One format is that of the application that created the data. The second is a translation of the application's formatting codes for boldfacing, justification, fonts, and so on into a generic form called **rich text format (RTF)**, which is recognized by all Windows applications. The third format is called **OEM (original equipment manufacturer)** text, which is used to paste text into DOS applications or when you prefer no formatting.

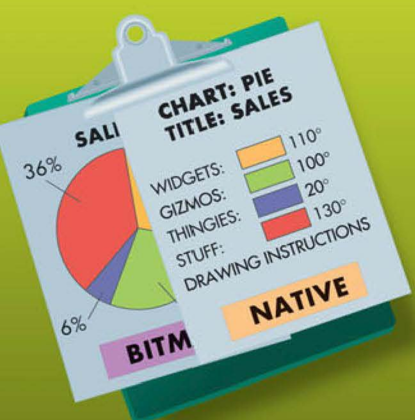
**3** For example, if the data is a graphic, Windows saves it in three formats—the original format, such as .TIF or .PCX; a bitmap format; and a metafile format. A **bitmap** is a record of the specific pattern of display pixels that need to be turned on to re-create the image in its original size. A **metafile** is a collection of commands that can be used by the' **graphic device interface (GDI) in Windows** to re-create the image. Metafiles are resolution-independent; that is, they aren't locked into a specific array of pixels, as a bitmap is. This lets metafiles take advantage of all the resolution your display or printer can provide, and it lets you resize images without distorting them. (A metafile is often called an **object-oriented** graphic because it is stored as a series of distinct objects–lines, rectangles, arcs—rather than as a map of pixels.)




**4** When you paste data from the Clipboard, the application receiving the data inspects the various formats in which the data has been copied. If you are pasting data into the application from which it was copied, the application will choose its native format.

**5** If you are pasting from one application into another, the receiving application inspects all the formats saved on the Clipboard. The application is looking for formats it understands and for which one of them retains the most formatting information. For example, a metafile graphic is preferable to a bitmapped graphic because a metafile contains more detailed information that the receiving application can use to change the graphic's size or placement on a page.

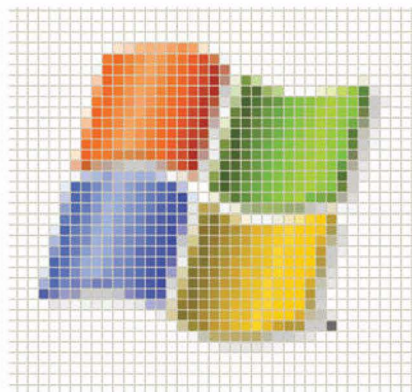

**6** To paste data that's in a format other than its native one, the receiving application first translates any information about the format of the data—such as boldfacing or fonts—into the formatting codes the receiving application uses. If it is receiving a metafile graphic, the application sends the commands contained in the Clipboard to GDI.exe and GDI32.dll in Windows, which control the graphic look of Windows XP and its ancestors. They, in turn, send the display driver the information the driver needs to create the graphic onscreen.





# How Vista Creates New Vistas

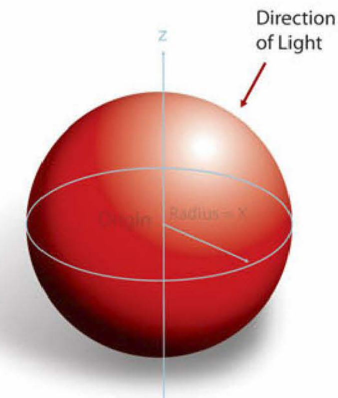
Over the years, Microsoft has determinedly made Windows more and more graphic. Sure, it's been a marketing ploy and eye candy, but it has been more graphic. The idea has been to create zipperless computing. With a sweep of the mouse or, now, a spoken command, you can find information, finish your work faster, and even rain hell on intergalactic invaders. But until Vista came along, Windows' and hotshot graphics were not a perfect marriage. Game players—the ones to whom superior graphics are so important, they make Photoshop users look like doodling cavemen—for a long time refused to leave DOS because it gave software direct access to hardware for snappy action. In Vista and with the Vista-only DirectX 10 interface, Windows has finally claimed an innovative approach to graphics that Microsoft is rightly proud of. To understand how Vista graphics work and why they are such an achievement, it helps to first take a look at how Windows graphics used to look.

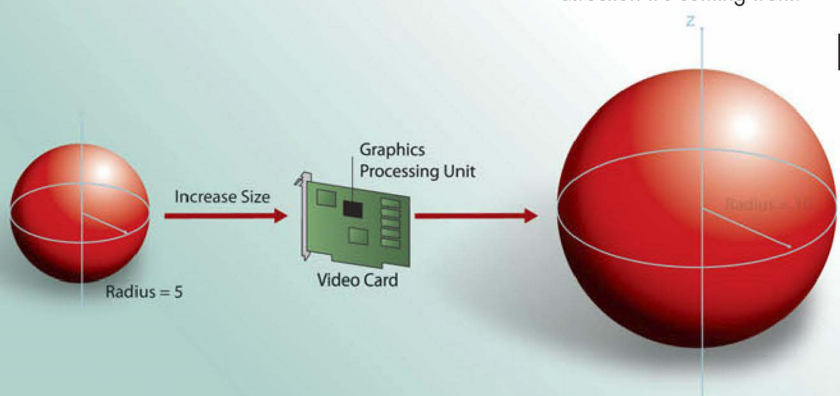


**1** Until Vista, the graphics used by Windows for icons, windows, thumbnails, wallpapers, logos—anything that contributed to the Windows **graphic user interface (GUI)**—were bitmaps. Every pixel in a bitmap is assigned a specific color and position *for that particular bitmap*. If the graphic changes sizes, moves, or changes any of its features, Windows has to rebuild it from scratch or load it from the hard drive. The process is a drain on the central processing unit and the result might only be a blocky approximation of what the graphic should look like.

**2** Even if you use graphics software to enlarge an image by careful interpolation to determine what colors are needed to create a smoother, truer enlargement, the result is imprecise.

**3** The graphics in all but the low-end editions of Vista are built of **vectors**. Instead of describing each pixel every time the graphic changes, Vista uses a mathematical description of the graphic. To describe a sphere, a vector graphic would include the radius of the sphere and another number representing the location of the center of the sphere. Another number would describe the color of the sphere. Still more numbers could tell Vista the intensity of the light striking the sphere and the direction it's coming from.

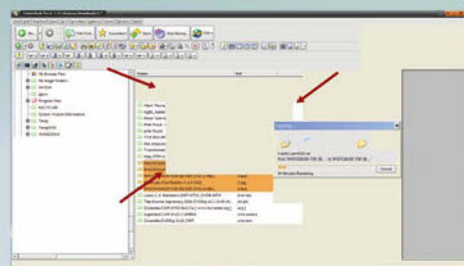



**4** Under Vista's new scheme for handling graphics, **Windows Presentation Foundation**, Vista does not have to determine the properties of thousands of pixels for a new image and store them in memory just to change the size of the sphere. The operating system only has to substitute a new number for the radius, and a **graphics processing unit (GPU)** on a video card renders the new ball.


**5** Several features that are new in Vista show the advantages and efficiencies of vector graphics. Once a vector image is created, it's a simple matter to resize and reuse it for other purposes. For example, pressing Alt+Tab in earlier Windows versions displays the standard icons for each application that is running. With Vista, Vectors make it a simple job to resize the actual, real-time screens of each program to use instead of static icons.

# How Vista Manages the Desktop

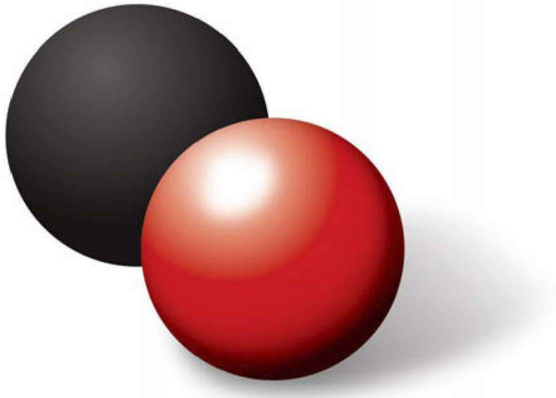


**1** Before Vista, when more than one element was on screen, Windows rendered only those parts of a dialog box, message box, or program that were "in the open." It didn't bother to render those parts of an application hidden by the displays of other programs. If you dragged a top-level program, Windows had to hurriedly calculate which pixels would be covered and which would be uncovered by the moving program and repaint the screen to add the pixels coming into play for the first time. Windows then had to write the location and appearance of the pixels going into hiding, so it could faithfully restore them later. When the display drivers could not keep up with the software, moving a program or dialog box could leave a cascade of identical dialog boxes or a sweep of empty white space

**2** Under the new **Desktop Windows Manager (DWM)** in Vista, each application sends DWM information about what would appear on screen if it were the only application and were taking up the entire screen. DWM stores this information in a **buffer,** dedicated in RAM, for each application.

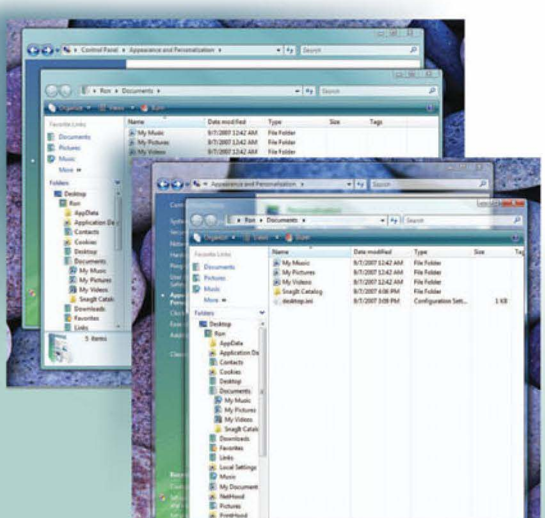
**3** When the information for all the screens has been gathered and stored, DWM uses it to construct the complete screen image, much like creating a montage from pictures clipped out of a magazine—parts of some pictures are covered by others, but they are not discarded. If the artist changes his mind, it's a simple matter to move the partly hidden clipping so it is fully exposed and obscures some other clipping.

# How the Aero Interface Sees All



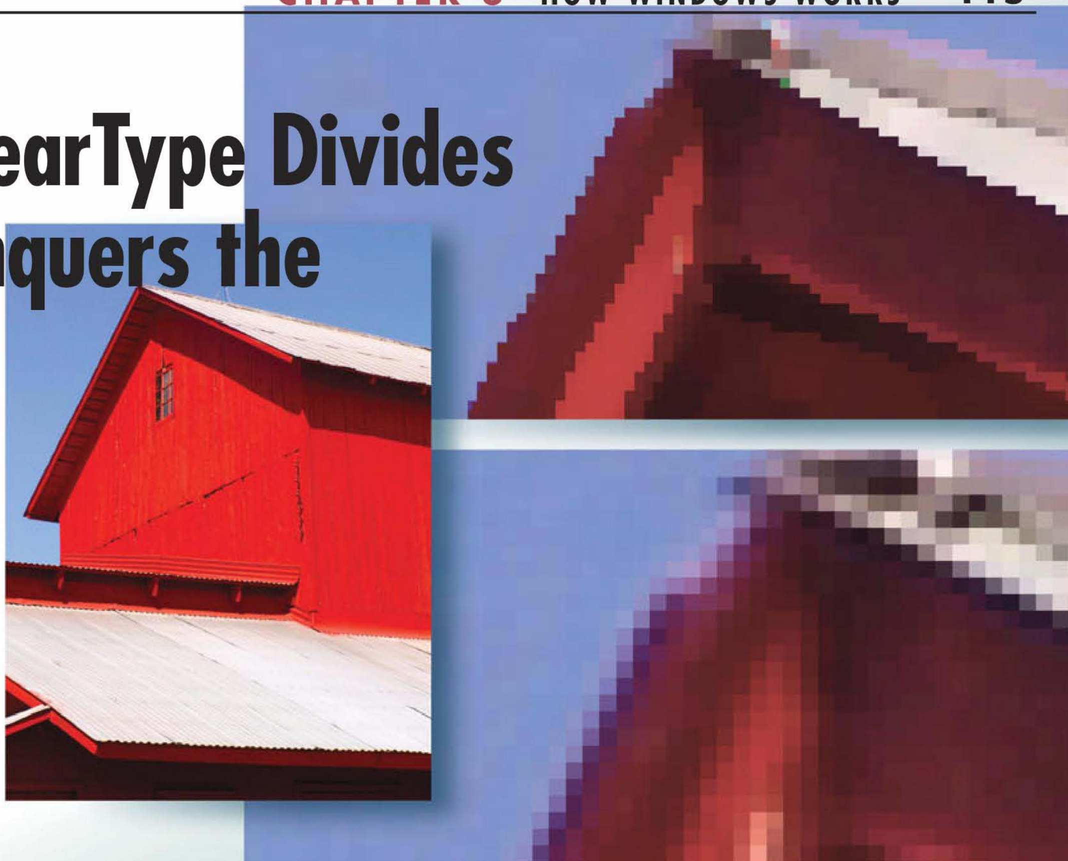
**1** Contributing to this new flexibility of onscreen elements are the pixels themselves—not the actual physical pixels we see on the screen, but the ***logical* pixels** the software sees. The logical pixels come with characteristics—**shadings**—that are evoked, not unsurprisingly, by **shaders**, specialized functions with which the GPU manipulates graphics on the pixel level. Returning to the example of a sphere created in vector graphics, it's as if the GPU shouted, "Glisten!" and the pixels changed their colors, saturation, and luminance so the sphere's surface looked as if it had just taken a dunk in water.

**2** Shading is not all the shaders' control. They add details, control movements on the pebble level, and enforce the laws of physics. Because shading calculates the visual effects on a pixel-by-pixel basis, the technique produces incredible detail, as shown in these two screen shots, one of which was rendered with older technology and the other with pixel shaders.

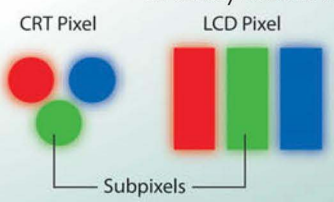


**3** Pixel shading is also responsible for the glass effects on the **Aero** desktop. Each element on the desktop is fully rendered, whether or not you can see it. All it takes to reveal, for example, a dialog box hidden behind the start menu, is for the pixels in the menu to activate a translucent shading.

# How ClearType Divides and Conquers the Jaggies



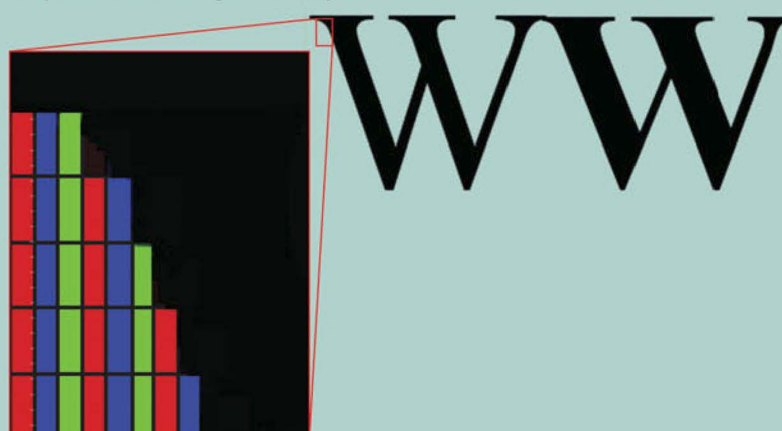
**1** Computer monitors create text and graphics on a **matrix**—a grid of invisible vertical and horizontal lines that define exactly where a display can light up a pixel. The problem with the matrix is diagonal lines and edges don't fit well, and the result is **jaggies**, the pointy shapes that appear whenever a distance along a diagonal line is smaller than the distance from one pixel to another. For example, here a diagonal edge formed by the barn's roof creates jaggies because the successively smaller divisions of the edge invariably become smaller than the screen's pixels. This is also called **aliasing**.




**2** One way to combat aliasing is with **anti-aliasing**, a trick that takes advantage of how the eye perceives shades of color, so that blurring the colors along a diagonal line actually makes the line look sharper. Anti-aliasing disguises the jaggies by using pixels along the edge that are a mixture of the two colors on either side of the line. The eye sees these pixels, not as separate colors dotting the intersection of sky and barn, but as a sharper drawing of the edge.

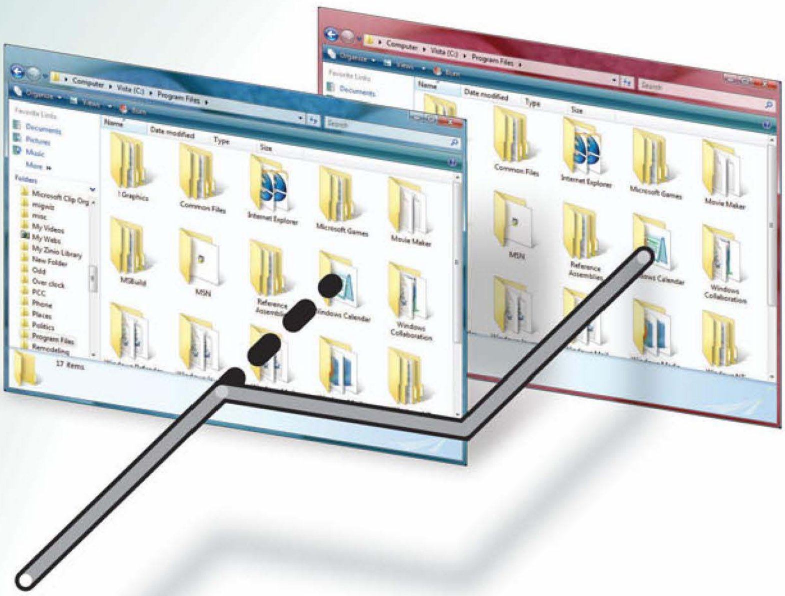
**3** For text, Vista carries the concept of anti-aliasing to the **subpixel** level, provided the text is displayed on an LCD screen. The technique, **ClearType**, does not work well with the old CRT monitors because their red, blue, and green subpixels form a triangle. The subpixels on LCD monitors are in horizontal rows.

**4** In addition to any anti-aliasing Vista may perform on the slanted strokes in text, ClearType also uses one or two of the three colors in each pixel to fill in gaps caused by aliasing. By turning off those subpixels, ClearType extends the black text to make minute changes in the shape so the text is easier to read. Without ClearType, the slant here would have had to follow the pixels outlined in white.



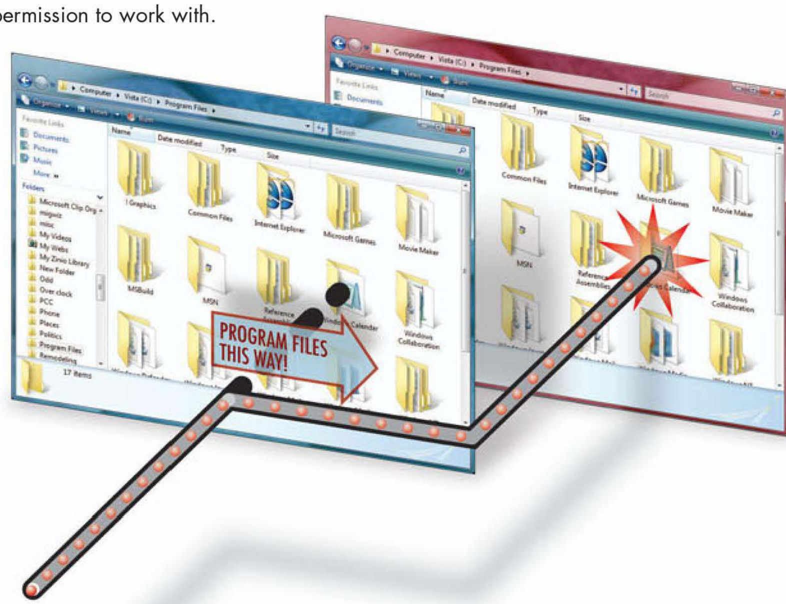
# How Vista Protects Your PC with Decoys

Under Windows XP, a person with **administrator** rights can run any program and create or change files anywhere in the computer. Ordinary users are limited to the programs they can run and the files they can see. Most users setting up their own computers understandably give themselves an administrator's **super user** permissions. Generally, software assumes administrator rights are in effect and that the programs can write to any file or any part of the' **registry of Windows**. This *de facto* XP standard also opens the door for viruses and spyware to enter the heart of the operating system, the **kernel**. With **User Access Control (UAC),** Vista closes the door and swallows the key.



**1** For each person using a computer, Vista's UAC sets up **file and registry virtualization**. User access control creates for each person a virtual system of directories and registry entries that mimic directories, such as *C:\program files*, and parts of the registry ordinary users do not have permission to work with.

**2** When a program tries to write a file to, for example, the *C:\program files* directory, Vista instead writes the file to the virtual version of *C:\program files*. Later if any program tries to read the file, Vista redirects the read request to the virtual file. To the program and the user, these switch-a-roos, like a TV episode of *Mission Impossible,* are indistinguishable from the real thing.




**3** With virtualization, any virus that attempts to access forbidden files is routed to the virtual directories; the virus does its dirty work and there's nothing to tip it off that it has not succeeded.


**4** File and registry virtualization is by itself not enough to protect a computer when a super user is at the keyboard. Under XP, malicious software is free to take advantage of the super user's access to sensitive areas. When a user logs in as an administrator, UAC registers two **tokens**. The first token authorizes the user to do anything an administrator normally does. The second token restricts privileges to those of a standard user.


**5** Once the login is complete, the second, restricted token is in control, restricting the user and all programs to approved actions.


**6** If an application—or a virus—wants to perform some restricted operation in a forbidden zone, UAC prompts the user for confirmation. (This may be the user's first clue that a virus is at work.) Once the user confirms the need for access to restricted operations, UAC carries out the action using the unrestricted token. Once the action is complete, it returns control to the restricted token.

**Halt! Who Goes There?**

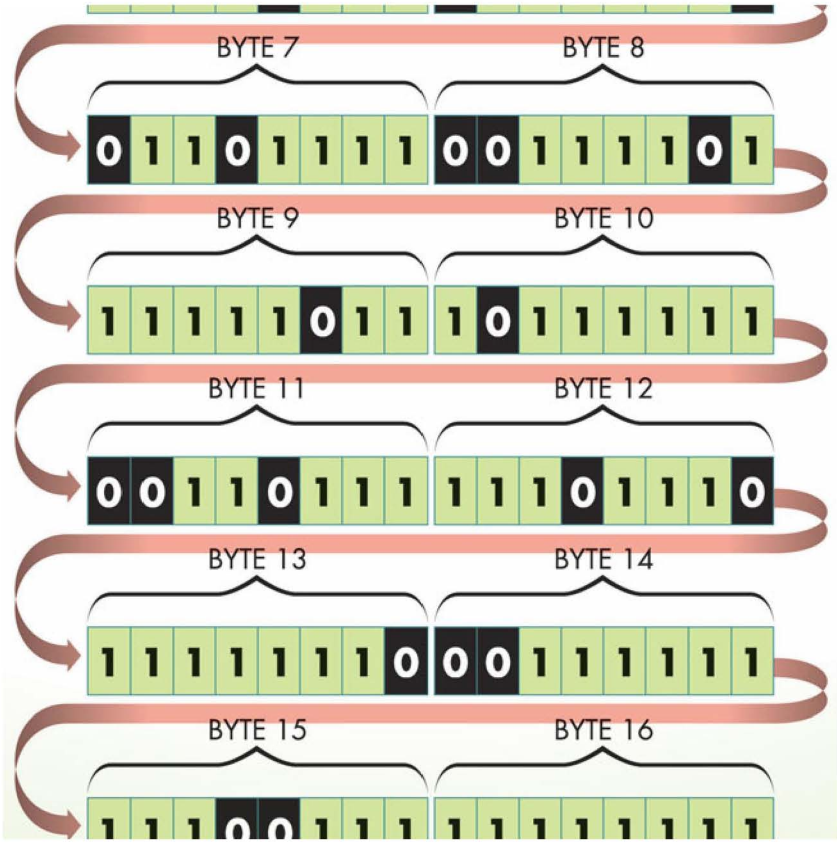
The tasks that trigger a UAC challenge include:

- Right-clicking an application's icon and clicking "Run as administrator"
- Changes to files in the system root or program files directory
- Installing and uninstalling applications
- Installing device drivers
- Installing ActiveX controls
- Changing settings for Windows Firewall
- Changing UAC settings
- Configuring Windows Update
- Adding or removing user accounts
- Changing a user's account type
- Configuring parental controls
- Running Task Scheduler
- Restoring backed-up system files
- Viewing or changing another user's folders and files

CHAPTER

9

# How Software Applications Do Your Work



**HERE'S** the quick definition of application software: It's everything except the operating system. Windows, DOS, OS X, UNIX, and other operating systems exist only so you can run applications—your **apps**. The long definition of apps encompasses everything from first-person shooter games such as *Bioshock* to the most strait-laced accounting program. Increasingly, the distinction between operating systems and applications is blurring as Windows, in particular, grows to encompass jobs that were once the purview of separate applications: faxing, file management, animation, videoconferencing, and most notoriously, browsing. But there are still applications whose power and scope exceed anything that could be packed into an operating system. In fact, if a job involves information, there's a program out there that will handle it.

What all software has in common is **data**. Data can appear in the form of stock prices, addresses, money, dates, statistics, phone numbers, recipes, probabilities, dots of color, pages of text, notes of music, the number of lives left in a game, positions in a virtual world, or your longitude and latitude on Earth. Data are raw facts. Names are data, and so are the letters that make up the word "names." Even computer code is considered data (which means data is manipulating data, a line of thought that is perilous to pursue). And all data are numbers, even if they're words or colors.

Computerdom has, by mutual consent, agreed on a standard called **ANSI (American National Standards Institute)**. In ANSI, each letter of the alphabet, punctuation, numbers, and even an empty space is assigned an 8-bit binary number. Eight bits is enough to represent all the decimal numbers between 0 and 255. A capital "A" is 65. A capital "B" is 66. A lowercase "b" is 98. An @ symbol is 64. $ is 36. ? is 63. 1 is 49, and 49 is 52 followed by 57. (Another path better left untrampled.)

All sounds can be expressed in numbers in the form of frequencies, amplitudes, and decibels. Your PC displays the colors on your screen by resorting to some combination of red, blue, and green—the only pure colors a display can create. With only 256 values representing 256 shades each of red, blue, and green, your monitor can produce more colors than most eyes can distinguish.

No matter what kind of data you feed into a computer, the PC ultimately sees it only as numbers, strings of zeros and ones written with transistors in the microchips of your motherboard (see Chapter 5). The decimal numbers of ANSI are converted to binary. A capital "A" becomes 01000001. A lowercase "b" is 01100010. After you master the concept that virtually anything can be represented as some collection of numbers—especially binary numbers—you have a good concept of what computing is made of.

Data, alone, though, is not really helpful. For example, 2.439, Donald Trump, green tells you nothing. It is not yet information. What software does is make data useful; it processes the data into information. No one program can process all kinds of data. Different raw data must be processed differently. Generally, software falls into one of these categories: database management, word processing, number crunching, graphics, multimedia, communications, or utilities. Each of those categories has more specialized subcategories:

- Database management underlies most other software, but it surfaces most visibly in inventory, mail list, contact management, and other software ranging from family-tree builders to digital jukeboxes.
- Word processing extends to desktop publishing and Web page design.
- Numbers—as numbers, not the ANSI kind—are crunched in electronic spreadsheets, accounting programs, and financial software.
- Graphics programs range from the precision technical drawing of computer-aided design (CAD) to the Paint program that comes with Windows.

- Multimedia includes anything that can be depicted best in pictures, music, sounds, and spoken words, whether an encyclopedia such as Encarta or a game such as *Call of Duty 4*.
- Communications software, once the province of thorough dweebs who never left the house, has been made the most vigorous category because of the Internet, electronic commerce, email, and broadcasting.
- Utilities are programs that help other software run better. They speed up drive access, clean up outdated files, and compress files. Utilities are truly geek software, and they are not discussed in this chapter, but you can find out how file compression works in Chapter 12. Multimedia and communications are covered in more detail in Parts 6 and 7.

## Database Managers

Software's reason for being is juggling. Whether the data are words, facts, or numbers, software finds new ways to compare, sort, order, merge, separate, and connect it—all an impressive juggling act. Collect a lot of data in some organized way, and you have a database. The software used most often to do that is called a **database manager**.

If you've visited a library that still uses card files, you've worked with one type of non-computerized database: one that illustrates perfectly the advantages of a computerized database. Typically, libraries have three sets of 3×5-inch card files: One set includes a card for each book in the library, sorted alphabetically by title; a second set also has a card for each book, but sorted by authors' names; and the third set is sorted by subject matter. That arrangement makes it easier for library patrons to find a book if they have only one piece of information about it; however, it is a terrible duplication of resources. The same information is repeated on cards in each of the three files. Obviously, it would be simpler and more efficient to have only one set of records that you can search by title, name, or subject matter. And that's exactly the advantage of a computerized database: It stores data that can be accessed and manipulated in many different ways.

Many programs that we don't think of as databases actually involve some form of database management. One reason why electronic spreadsheets such as Lotus 1-2-3 and Excel have become so popular is that, in addition to being capable of calculating complicated mathematical formulas, they can sort and extract both mathematical and textual data. Accounting and inventory programs are specialized databases. Even word processors use database features in their spelling checkers and mail-merge operations. But the databases we'll look at here are those dedicated exclusively to the collections of data. The range of tasks a database manager can perform varies with the complexity of the program. But in general, they all do these jobs:

- Database managers let you define a **data type** for the information you want to store; alphanumeric or numeric, for example. They also define a data format that aids in retrieving and organizing the data. A single piece of data might, for example, be limited to a certain length or to specific values.
- Database managers store data in records. A **record** is a collection of data about a particular person, place, or thing. The individual items are contained in **fields**, similar to the blank areas on a paper form. A record, for example, could include fields for information on one person's name, address, and phone number. The records are displayed in an onscreen form, where field data is entered, edited, deleted, and viewed. Several records with the same fields of information for entries constitute a **table**.

- Database managers carry out **queries**, which are searches for data that meet specified criteria, to allow you to retrieve certain subsets of the data. These queries **sort** and **filter** the data to let you see it from different angles. A typical query, for example, would ask the database to display all records for bills that are more than 60 days overdue.
- Database managers perform calculations on the data. Not only do they do mathematical calculations, they also perform "if true" **logic** tests—if inventory is below X amount, order more—and **parse** text data, allowing you to perform operations such as combining a person's first and last names from two pieces of data into one.
- Finally, database managers present the data in a formatted, easy-to-read **report**, which can include **calculations** on the data, such as the total value of sales orders taken from several store locations.

How database managers perform these operations varies with the type of database. Many databases are designed for simple on-the-fly use. They provide ready-made tools and commands for performing relatively simple, common operations, such as searching and sorting. More powerful database managers, such as Microsoft Access, Corel Paradox, and Lotus Approach, include their own programming or script languages that let the managers be used to create other specialized applications designed for specific tasks, such as maintaining a wine list or handling personnel records. A person might use an inventory program without realizing it was created in the Paradox programming language.

Database managers are distinguished from one another on another basic level. Some can manipulate only one collection of data—a **table**—at a time; these database programs are called **flat-file** database managers. Other managers can link data from several different tables. They're called **relational** database managers because they define relationships among common elements of different tables.

In this chapter, we'll look at some of the fundamental operations performed by database managers: storing data; creating **indexes** that allow fast retrieval of data; and, in the case of relational database managers, **linking** data from different tables.

## Spreadsheet Software

A program called VisiCalc is responsible for the fact that you're using a personal computer today. The early PCs—by today's standards, wimpy machines—made by Radio Shack, Apple, and Commodore, were mostly the playthings of electronics hobbyists. The mere fact that these hobbyists could buy a functioning, real computer was enough in itself to satisfy their experimental urges. The computer didn't have to do anything really useful, as the big mainframe computers did.

Then, two young men named Dan Bricklin and Bob Frankston wrote a software program called VisiCalc, and the direction of the personal computer changed radically. Suddenly there was a real, practical reason to buy a PC. Running on the Apple II, VisiCalc was an electronic version of what accountants, department managers, bankers, and financial officers had been staring at on paper for ages: the spreadsheet. In both its paper and computerized forms, the spreadsheet is simply a grid of vertical and horizontal lines into which labels and numbers can be entered to keep track of all sorts of numerical records, usually financial. But there is an important difference between the paper and electronic versions. On paper, to enter the sum of a long column of numbers, you must add them manually

or with the aid of a calculator, and then pencil in the result. If any numbers in the column change, you must pull out the calculator and do the whole job over again.

With an electronic spreadsheet, you can enter a formula that represents the sum of that same column of numbers, and the computer program will do the calculation and enter the result in the proper position on the grid. If the numbers change later, the electronic spreadsheet will automatically recalculate the formula and enter the new sum.

If the spreadsheet did nothing more than such simple calculations, it would still be a great saver of time and labor. But it actually does much more complicated tasks. Complex mathematical formulas are reduced to a few formula commands. Also, various **cells**—the rectangles formed by the grid's lines—can be linked so the results of one calculation become an element in a formula elsewhere. A change in the data in one cell can cascade in seconds through hundreds of calculations scattered among thousands of cells.

Imaginative business people quickly saw how this new computer program was a real business tool that justified the cost of personal computers, making microcomputers machines for the office rather than merely toys for hobbyists. The demand for PCs created a market for programmers, who in turn created database managers, word processors, and other software that generated, in an upward spiral, more demand for PCs.

Early electronic spreadsheets were significant in another way. They represented the first **graphic user interface (GUI)**, a term reserved today for the way that Windows and Macintosh systems display programs. Compared to today's GUIs, the first electronic spreadsheet was a crude but recognizable onscreen representation of the paper ledger that people were already used to. That fact made it simple for computer users to grasp how the electronic spreadsheet was supposed to work. That capability to mimic real-world tools continues to be one of software's greatest strengths. One historical note: No one uses VisiCalc today. Although it broke important ground in the development of the PC, VisiCalc failed to keep up with the innovations of other electronic spreadsheets, such as Lotus 1-2-3, Excel, and Quattro Pro, which added database and graphing capabilities to VisiCalc's number crunching. Similarly, the software you find most useful on your PC today will probably look ridiculously primitive in another decade.

In this chapter, we'll look at two of the fundamental operations of electronic spreadsheets: how they arrange all the data in their cells so that cells can be linked to one another, and how formulas are used to perform the spreadsheets' calculations.

## Word Processors

The office personal computer quickly grew from a number cruncher to a **word processor**—not surprising when you consider how the business world most often communicates—with words. Whether through a company policy booklet, a letter to a customer, an annual report, or a memo to tell some boss to take this job and shove it, people in business communicate in written words, and they want those words to be perfect.

A sheet of paper that's a crazy quilt of whiteout and penciled corrections might be an accurate record of words, but by today's computer-perfect standards it simply is not acceptable. Nor should it

be. Word processors provide tools not just to record and print words but to make them attractive and more *communicative*; and not with *words alone*, but with photos, graphics, *and color*.

In this chapter, we'll look at the function most crucial to this task: formatting text onscreen in preparation for sending it to the printer.

## Graphics Software

We think of the Mona Lisa as a brilliant example of Renaissance art, as a mysterious image, or as a thoughtful study in composition, light, and shadow. We don't think of it as a mathematical formula. But in the computer world, all art, graphics, shapes, colors, and lines involve some type of mathematical **algorithm**. That statement isn't meant to belittle the works of Da Vinci and other great artists. Mathematical algorithms cannot create art; that still takes a true artist, whether the artist's tools are brush, oils, and canvas or a computerized stylus. But math embedded in specific file formats can describe any piece of existing art. A graphics-file image of the Mona Lisa that you can display on your PC is the result of mathematical calculations on the bytes of data saved in that file.

Today, all the capabilities of a darkroom and an artist's studio are available on a personal computer. You can retouch, lighten, darken, crop and do virtually anything else to a computer image that you could do to a photograph. You can do more, really. With a PC, even a semi-skilled retoucher can make Mona Lisa frown. Other software lets an artist re-create the effects of different media: oils, watercolors, airbrush, charcoal, and so on. The artist can even mix media in ways that are difficult to match in real life—to obtain, say, the effect of a water-based paint dissolving chalk lines on paper.

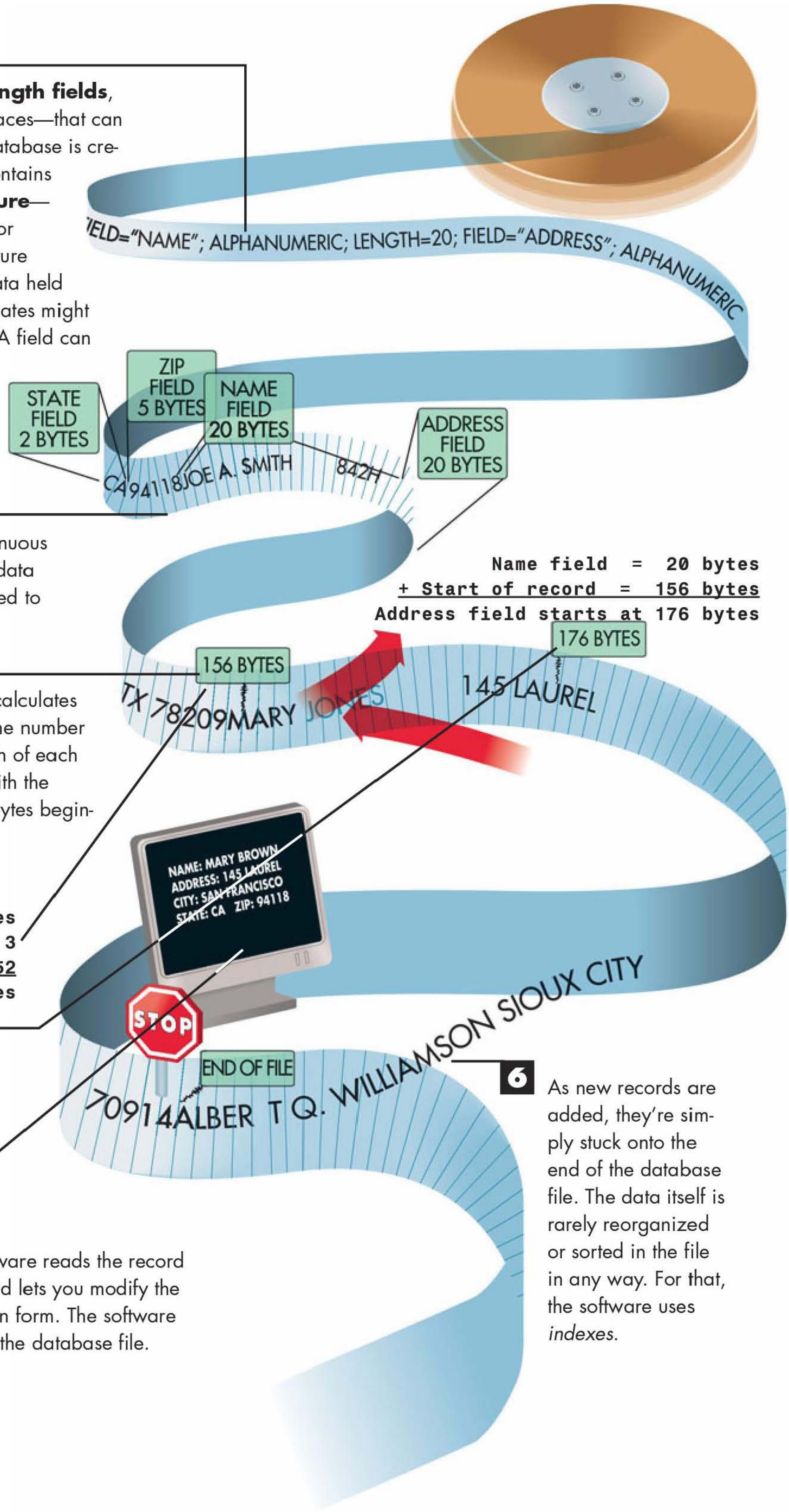
Modern PC graphics are not just about creating pretty pictures. We live in a world of images and colors; not a world of words, which are, after all, only abstractions for the things we see, feel, touch, and do. We use shape and color in everyday life to convey information faster than words can. Just consider the red, octagonal stop sign; you can be illiterate and know what it means. So, too, in computers, graphics are an increasingly important way of conveying information. Imagine an onscreen map of a city, showing in red those Zip code areas where family income is more than $100,000. You comprehend it at a glance. A spreadsheet printout of the same information consisting of nothing but numbers would require ponderous study to comprehend.

Shapes and colors are information as surely as words are. The difference is that words are nicely defined and limited, but shapes and colors have infinite variations, which means that a PC handling graphics is up against a more daunting job than the old text-based DOS computers. There are two basic ways in which graphics software copes with this infinity of variation: through bitmap and vector graphics. This chapter looks at how bitmapped and vector images are stored and how graphics software translates the data into images.

Shapes and colors are information as surely as words are. The difference is that words are nicely defined and limited, but shapes and colors have infinite variations, which means that a PC handling graphics is up against a more daunting job than the old text-based DOS computers. There are two basic ways in which graphics software copes with this infinity of variation: through bitmap and vector graphics. This chapter looks at how bitmapped and vector images are stored and how graphics software translates the data into images.

# How a Database Organizes Information

## Fixed-Length Field Records




**1** Most database information is stored in **fixed-length fields**, so called because the number of characters—spaces—that can be used for each field is determined when the database is created. The beginning of a fixed-length field file contains information that defines the file's **record structure**—each field's name, **data type** (usually numeric or alphanumeric), and length. In addition, the structure might include information on the format of the data held in the field; for example, a field used to record dates might require the MM-DD-YY (month-day-year) format. A field can also be required to **validate** the information entered into it; for example, if the data entered into a validating state field is not one of the 50 postal service abbreviations, it will be rejected.

**2** The rest of the file is data, laid down in one continuous stream. The locations at which specific pieces of data are recorded are determined by the lengths allotted to each field.

**3** To find any given record, the database software calculates the location's **offset** through a simple formula: The number of a particular record multiplied by the total length of each record equals the starting point of that record. With the starting point calculated, the program reads the bytes beginning at that point in the file.

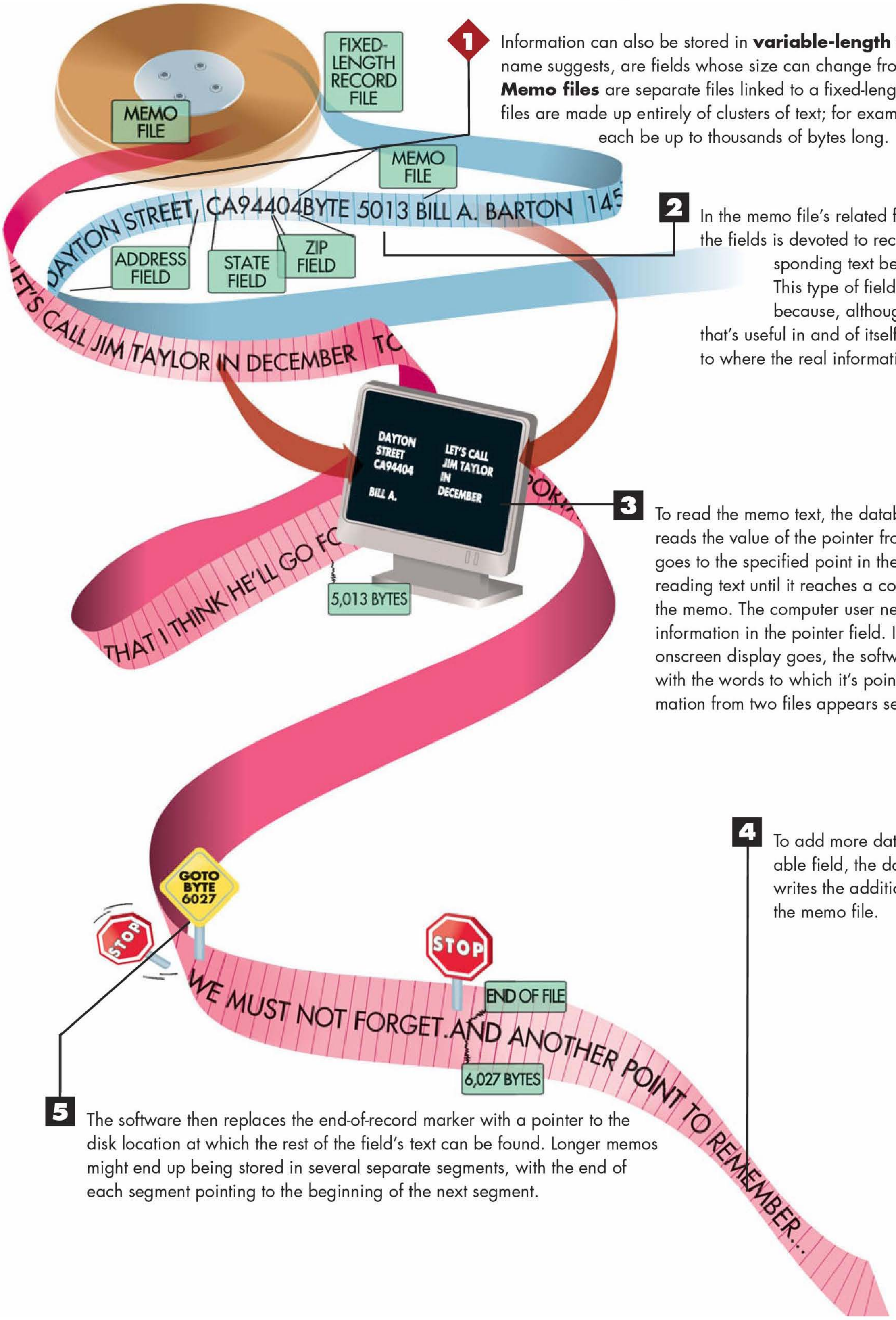
```
Record length  =  52 bytes
      Record number  =  3
        Minus 1  =  2 x 52
Record 3 starts at 104 bytes
```

**4** To locate fields *within* each record, the program follows a similar process of calculating the number of bytes in the fields preceding the field it wants to find, and then reads bytes at the field's starting point.

**5** To **modify** an existing record, the database software reads the record into variables it creates in the computer's RAM and lets you modify the information in these variables through an onscreen form. The software then writes the new contents of these variables to the database file.

**6** As new records are added, they're simply stuck onto the end of the database file. The data itself is rarely reorganized or sorted in the file in any way. For that, the software uses *indexes*.

# Variable-Length Field Records




**1** Information can also be stored in **variable-length fields**, which, as their name suggests, are fields whose size can change from one record to another. **Memo files** are separate files linked to a fixed-length database file. Memo files are made up entirely of clusters of text; for example, memos, which can each be up to thousands of bytes long.

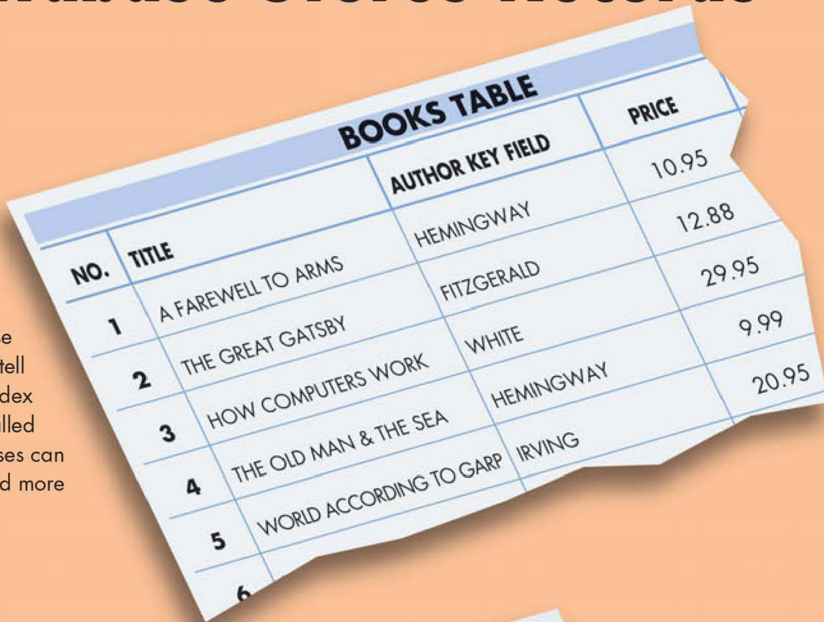
**2** In the memo file's related fixed-length record, one of the fields is devoted to recording where the corresponding text begins in the memo file. This type of field is called a **pointer** because, although it contains no data that's useful in and of itself, the field's contents point to where the real information can be found.

**3** To read the memo text, the database manager software reads the value of the pointer from the fixed-length file, goes to the specified point in the memo, and begins reading text until it reaches a code that marks the end of the memo. The computer user never actually sees the information in the pointer field. Instead, as far as the onscreen display goes, the software replaces the pointer with the words to which it's pointing. The combined information from two files appears seamless.

**4** To add more data to an existing variable field, the database software first writes the additional text to the end of the memo file.

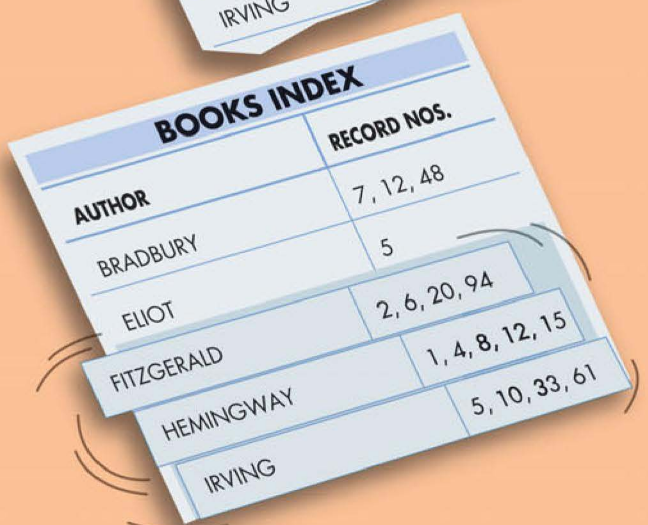
**5** The software then replaces the end-of-record marker with a pointer to the disk location at which the rest of the field's text can be found. Longer memos might end up being stored in several separate segments, with the end of each segment pointing to the beginning of the next segment.

# How a Database Stores Records




**1** To index records, the database manager first requires you to tell it on which of the fields the index is to be based. This field is called the **key field**. Some databases can have more than one index and more than one key field.

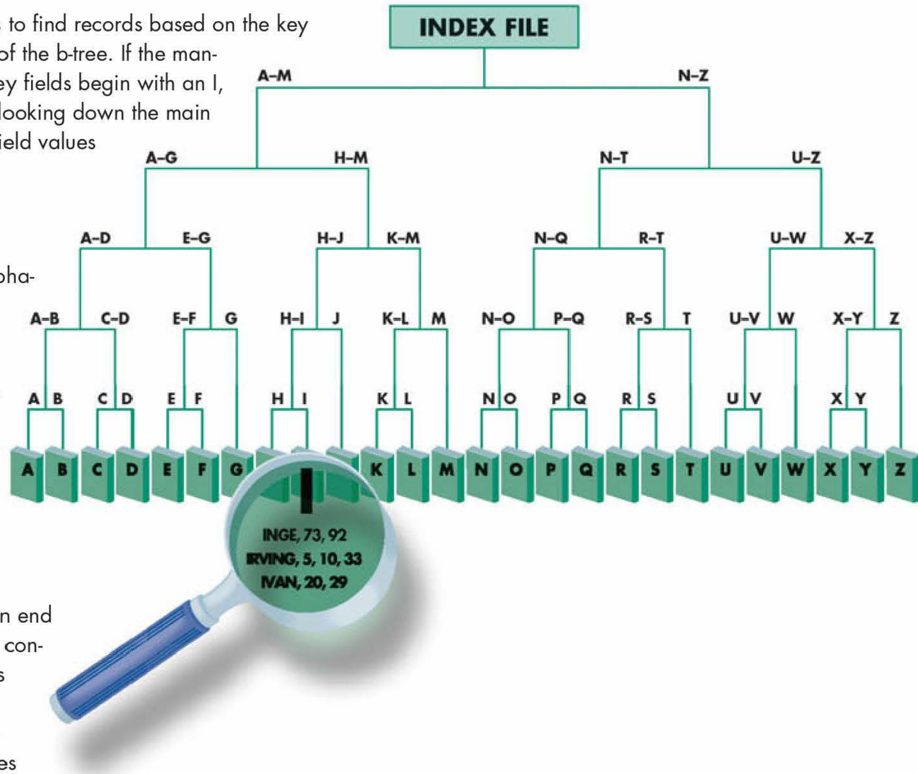

**2** The database manager reads each record and constructs a temporary file consisting of the values contained in each record's key field and corresponding pointers that give the location of each record in the database file. If duplicate values are found, each duplicate entry is recorded in the index file.




**3** After the database program has read all the values and their pointers, or record numbers, into the temporary file, it arranges the copied values in alphanumeric order, creating an *index*.

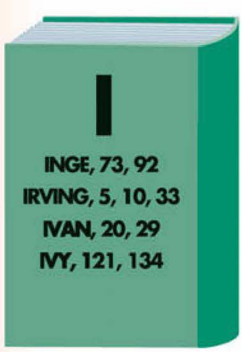
**4** The database writes the ordered information to an index file that is structured as a binary tree. The binary tree, or b-tree, is designed to speed up the process of finding information in the index file. It's an upside-down tree in which each node has two branches. These branches break logical divisions of the index file into increasingly smaller halves. For example, A–M represents one of the first two branches of the tree and N–Z represents the other main branch. A b-tree search lets a database search a million-entry index by checking only 20 sets of nodes rather than each of the one million nodes.

**5** When the database manager needs to find records based on the key field, it checks successive branches of the b-tree. If the manager is looking for records whose key fields begin with an I, for example, the manager starts by looking down the main trunk of the tree, where it finds key-field values beginning with A–M.

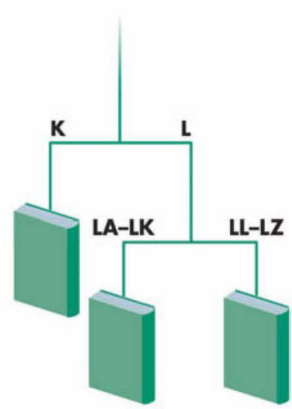



**6** Because I comes before M in the alphabet, the manager next looks at the key-field values halfway between A and M. There, it finds values beginning with G. I comes after G, so the manager looks halfway between G and M, and so on, until it finds values beginning with I.

**7** Eventually, the manager arrives at an end node—the leaf, so to speak—which contains a short, fixed number of entries (eight or so, depending on the program) and their pointers. It finds the entry it has been looking for and uses the pointer to locate the actual record in the database table.



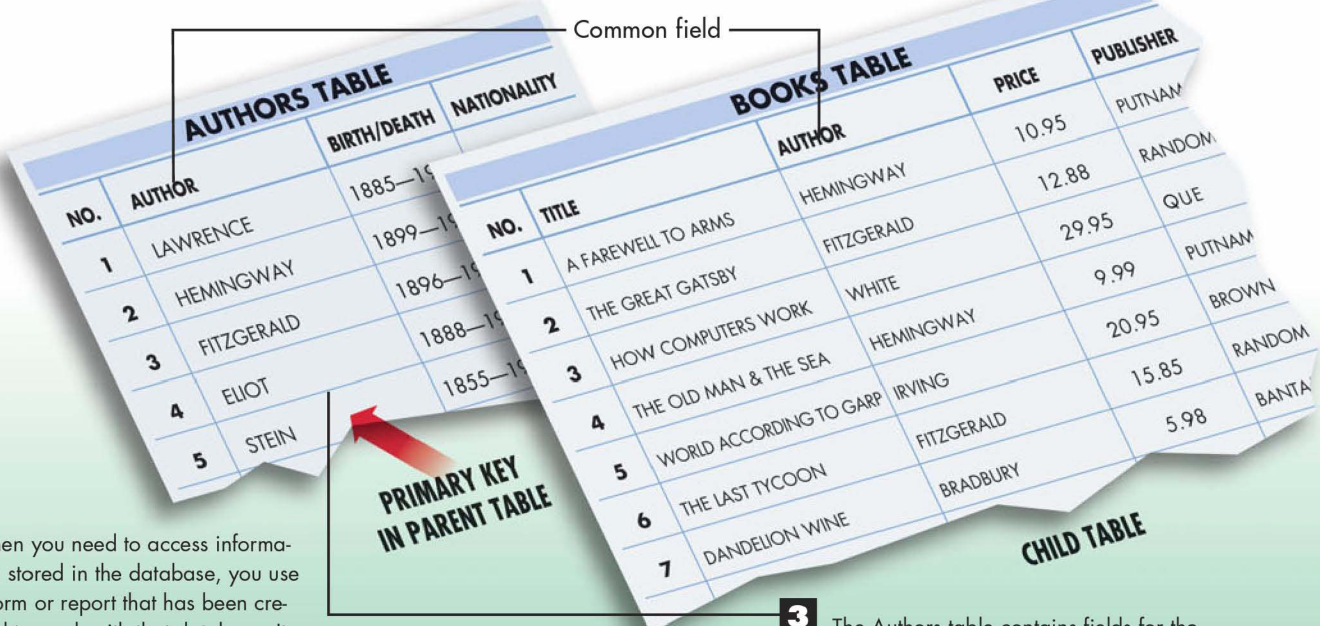

**8** To re-index the database after new records have been added to the database, the program puts each new index entry into a blank space under the proper "leaf" in the index's b-tree.




**9** If there's no room under the leaf, the software creates two new nodes under what had been the last node. For example, an L node would be divided into an LA–LK node and an LL–LZ node, each of which would receive roughly half of the parent node's information.

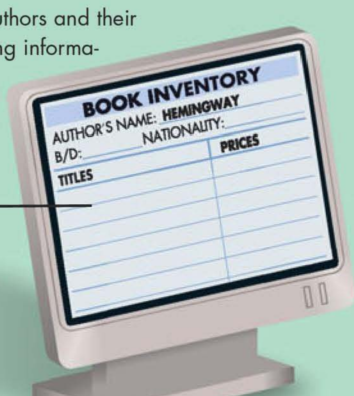
# How Databases Make Connections

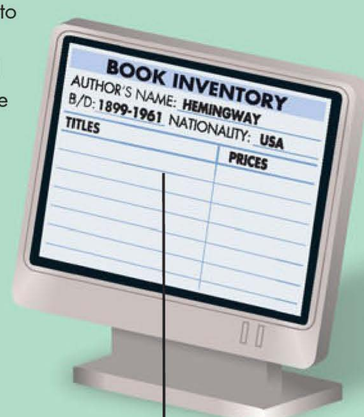
**1** Imagine moving to a new city and filling out a single form that automatically updates information about your new address and phone number on your driver's license, in the phone book, all your subscriptions, your bank records—everyone you deal with. That's the concept behind a **relational database**. It separates information into **tables** that share each other's data. A change of information in any one of the tables is picked up by any other table that needs it. A relational database is designed so that most information must be entered only once, and yet it can be used by more than one table of data.




**2** When you need to access information stored in the database, you use a form or report that has been created to work with that database. It displays the fields that hold the information you want to retrieve. In this example, you want to obtain information about authors and their books, combining information from two tables.

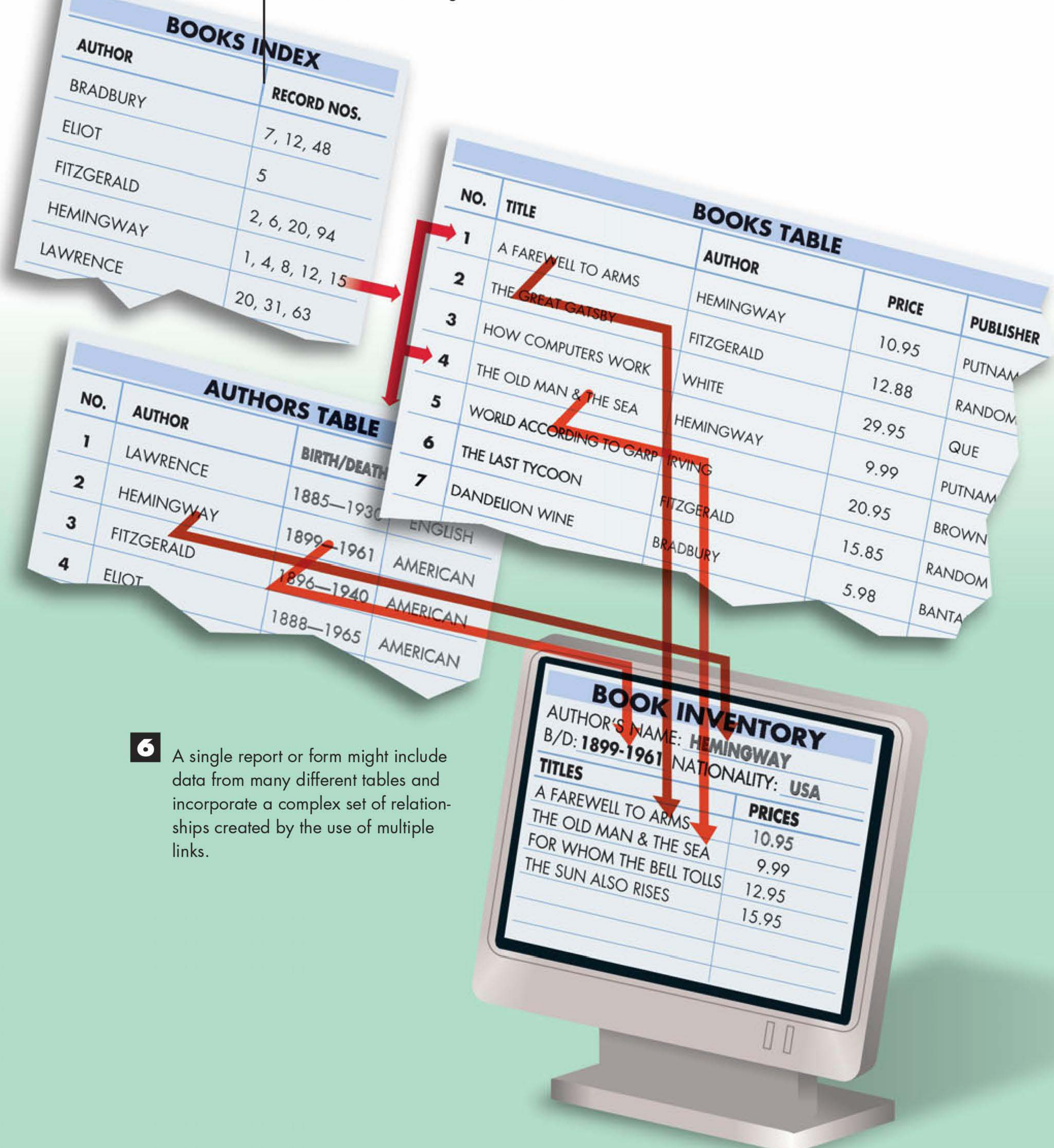
**3** The Authors table contains fields for the name, nationality, year of birth, and year of death of each of several authors. The Books table contains only the titles, publisher names, prices, and author names for books. The Author field in the Authors table is the primary key for the relationship between the tables. A **primary key** must be a unique field in the parent table; that is, it must identify only a single record in that table. The Books table in this example is called the **child table**. The Author field is also a part of the Books table, but in that table, the contents of the Author field are not unique: The same name can appear in several records. (One way to remember these terms is to recall that a parent table can be related to several children, but a child table has only one set of parents.)




**4** The **form** for this example specifies several fields from the Authors table, so the software finds that table, pulls out the contents of those fields for the current record, and displays them onscreen.

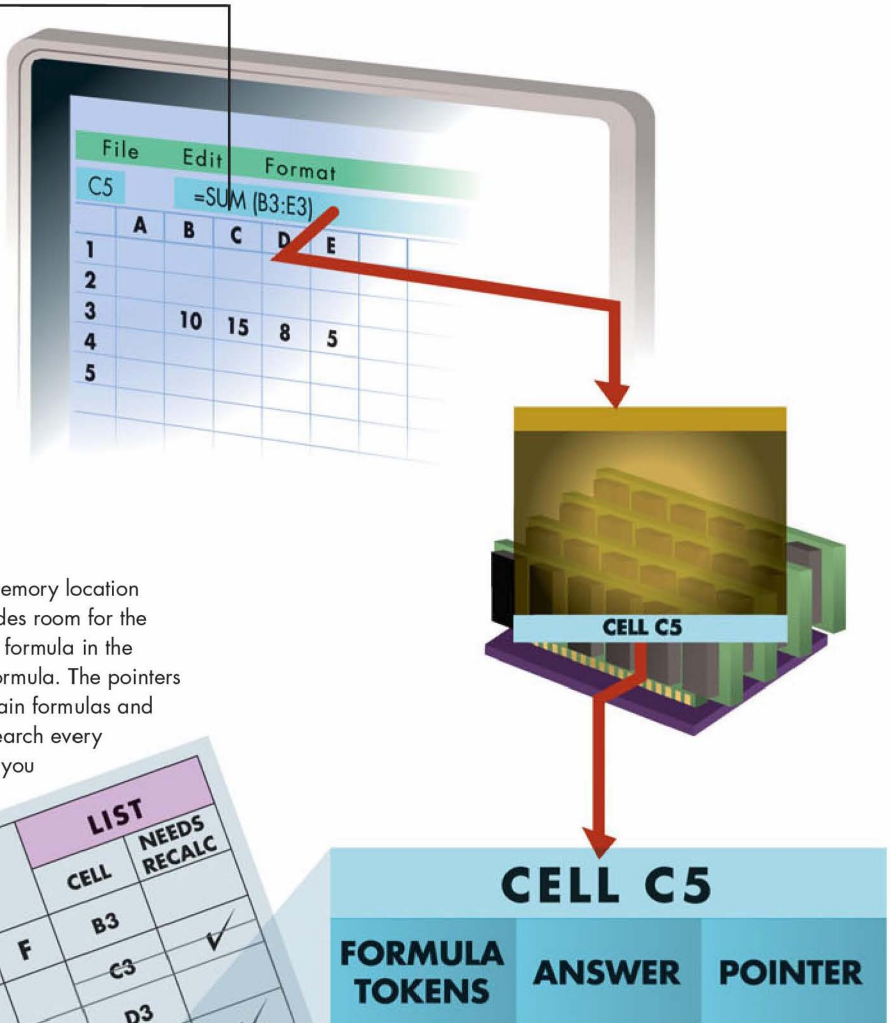
**5** The database manager then goes to the index of the Books table to find all the records with "Hemingway" in the Author field. It uses the pointers of those index entries to locate the right records in the Books table, and then it pulls out the requested fields and displays those onscreen. When you switch to a new record in the Author table (for example, Fitzgerald), the software displays that information from the Author table, and then goes back to find all the records in the Books table that have Author fields matching the new record.



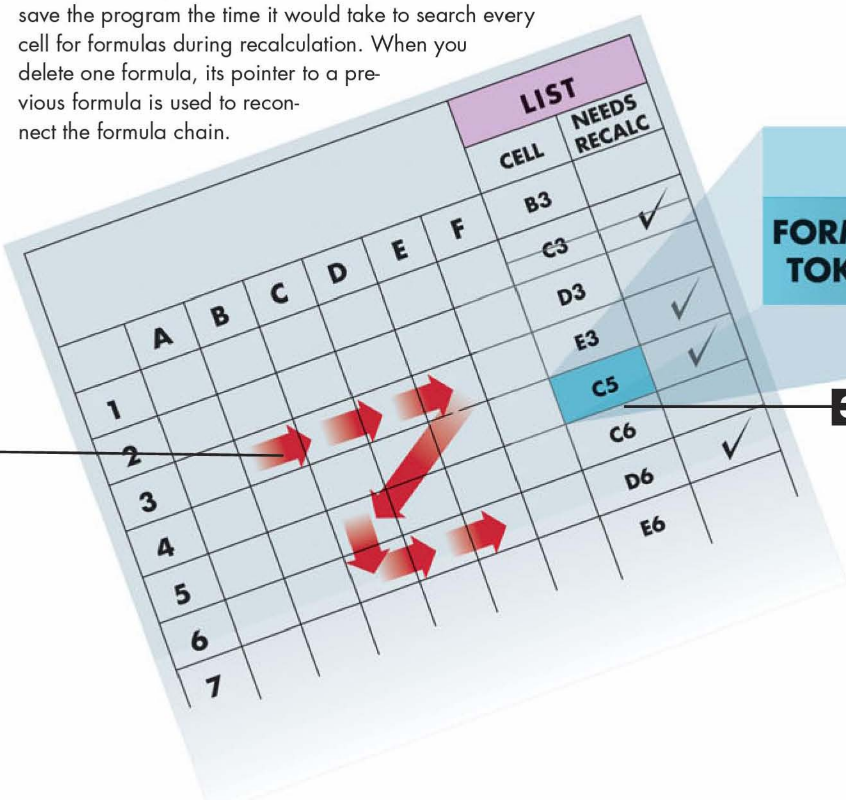

**6** A single report or form might include data from many different tables and incorporate a complex set of relationships created by the use of multiple links.

# How Spreadsheets Solve Formulas

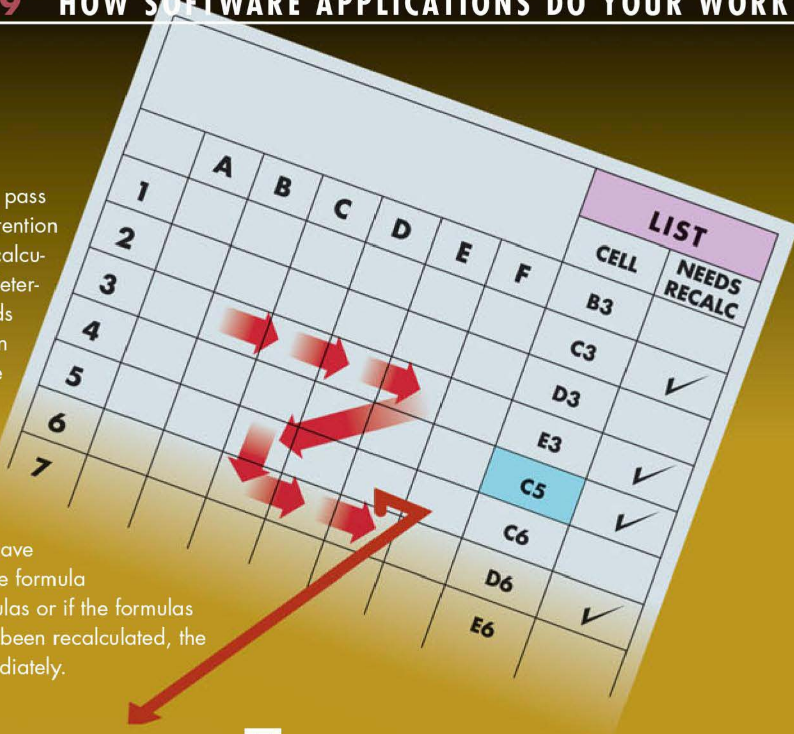
**1** When you type a formula into a cell, the spreadsheet processes the formula through a **minicompiler** that converts the function names into a more efficient, tokenized format, in which functions are represented by specific numbers. For example, functions such as SIN and COS are converted into specific byte values the spreadsheet recognizes as meaning sine or cosine. The compiler also stores the formulas in **reverse Polish notation**, so that, for example, (3 + 2) * 10 becomes 3 2 + 10 *. This type of notation is more efficient in terms of both space and speed.



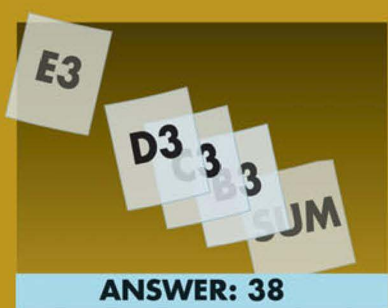

**2** The result of the compilation is written to a memory location reserved for that cell. The location also includes room for the result of the calculation, a pointer to the next formula in the spreadsheet, and a pointer to the previous formula. The pointers create, in effect, a list of those cells that contain formulas and save the program the time it would take to search every cell for formulas during recalculation. When you delete one formula, its pointer to a previous formula is used to reconnect the formula chain.




**3** When the spreadsheet is recalculated, the program saves work and time by making a first pass through the list created by the pointers of cells that contain formulas. It finds those formulas that depend on data that has changed, and marks each one that needs to be recalculated.

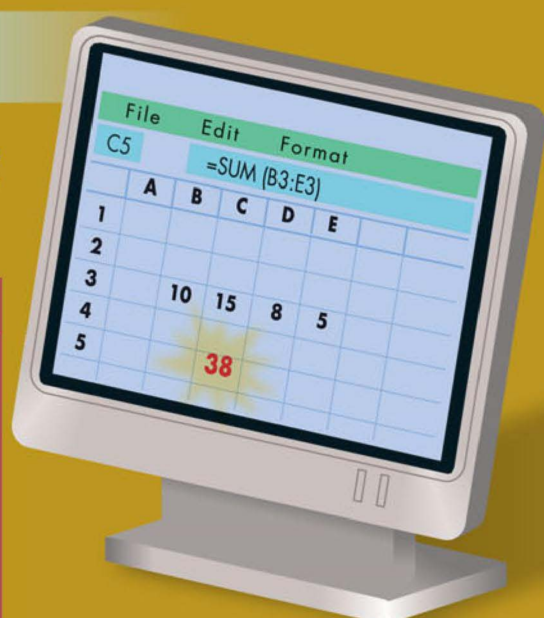



**4** The program then makes a second pass through the list, this time paying attention to only the formulas marked for recalculation. For each, the spreadsheet determines whether that formula depends on another formula that hasn't been recalculated yet. If so, it adjusts the cell's pointers and the pointers of connected cells so that the dependent formula moves to the end of the list. (This process pays off the next time the spreadsheet is recalculated—the program won't have to change the pointers again.) If the formula doesn't depend on any other formulas or if the formulas on which it depends have already been recalculated, the software recalculates the cell immediately.




**5** To calculate a formula, the spreadsheet software feeds the data the formula and the formula codes requested into a **calculation engine** that generates the answer and writes it to the part of memory allocated to hold information for that cell.

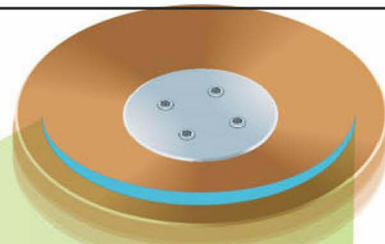
**6** The spreadsheet then moves to the next formula and repeats the process until it ends by finally calculating those formulas (earlier placed at the end of the list) that are dependent on other formulas.


**7** In some spreadsheet programs, such as Excel, the software updates each cell onscreen as soon as it has been calculated. Other spreadsheets wait until the whole spreadsheet has been recalculated before updating the display.

### Auto Recalc

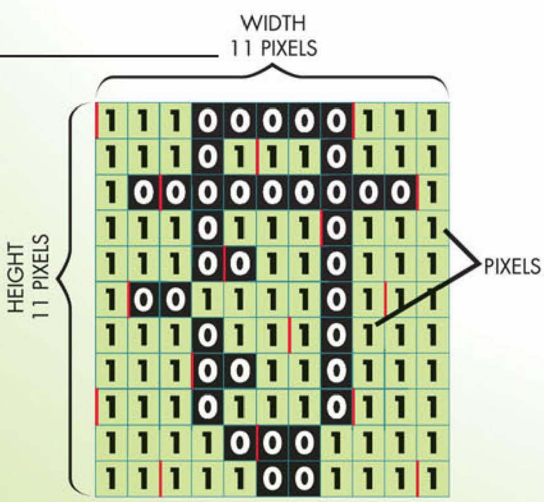
If you've turned on a spreadsheet's automatic recalc feature, the spreadsheet is updated every time you make a change that affects any formula. How does it do this? When you create a formula, the spreadsheet software marks all the cells on which that formula depends by changing a notation in each of their records. In addition, it leaves itself hints in those cells about how to find the formulas that depend on them, a more efficient method than using pointers. When you make a change to any cell that's been marked in this way, the software finds the formula or formulas that are affected and recalculates them.

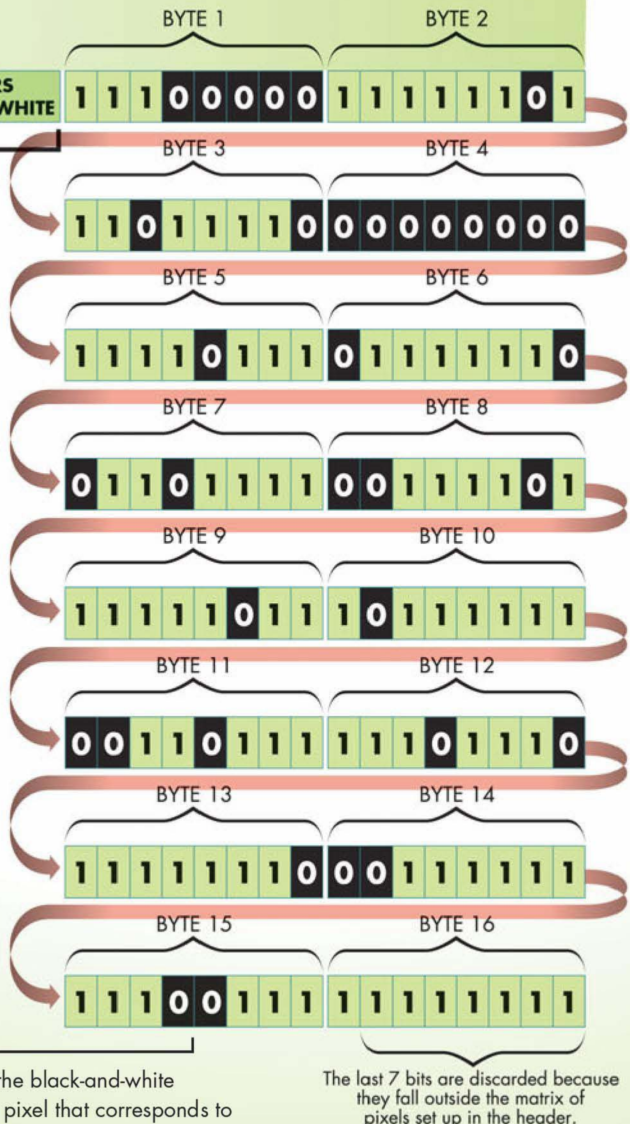


# How Bitmaps Save Artwork

**1** When a graphics program reads a **bitmapped** file from a drive, it first looks at information contained in the file's *header*, which is several bytes at the beginning of the file that contain information the program needs to interpret the data in the rest of the file. The header begins with a **signature** that identifies the file as a bitmap. You don't see this signature, but you can tell that a file is a bitmap if it has an extension such as .BMP, .PCX, .TIF, or .JPG. Following the signature, the header tells the width and height of the image in **pixels**, which are distinct points of light, and then defines the *palette* (how many and which colors are used in the image).


**2** After determining the parameters of the graphic file, the program reads the bytes of data following the header that contain the image as a pattern of bits. The simplest bitmapped image has only black-and-white pixels. For images of this type, the graphics program needs only two pieces of information: the location of a pixel and whether to turn the pixel on or off. The locations of the pixels are determined by the image's width and height as defined in the header. In this crude image of a man in a hat, the line pixels wrap every 11 bits to the next row of pixels.

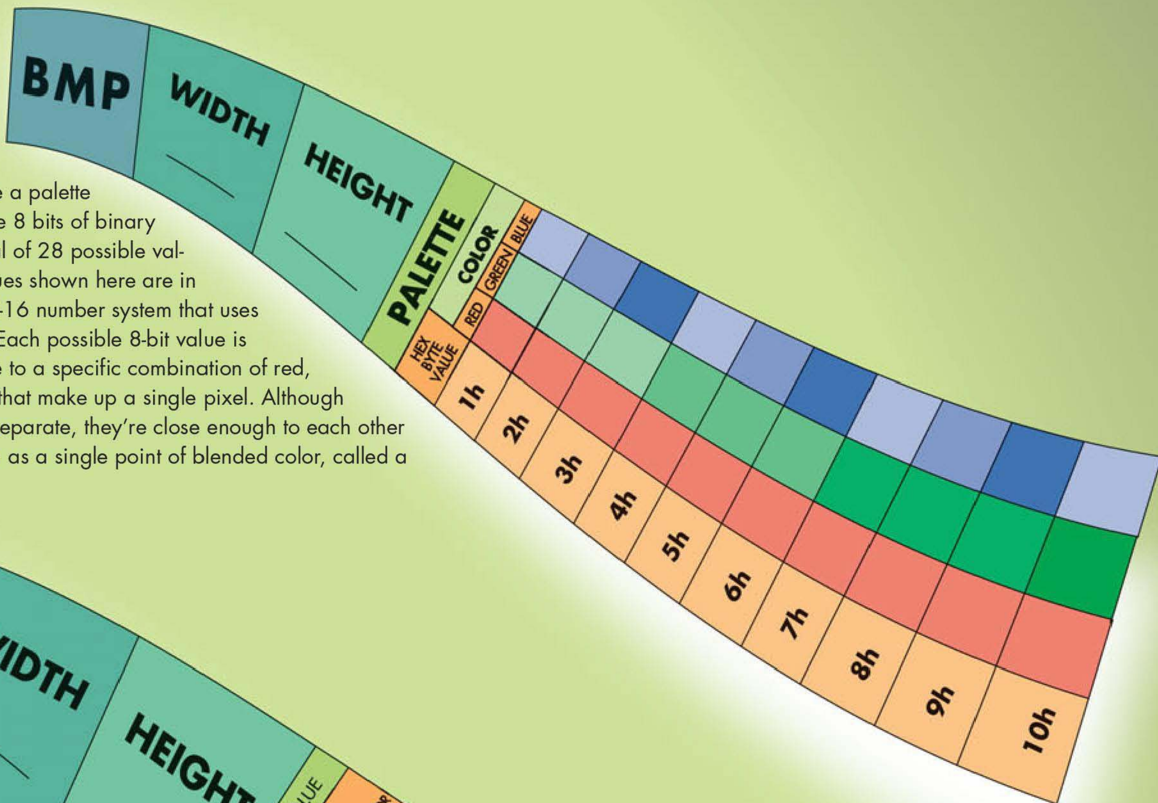


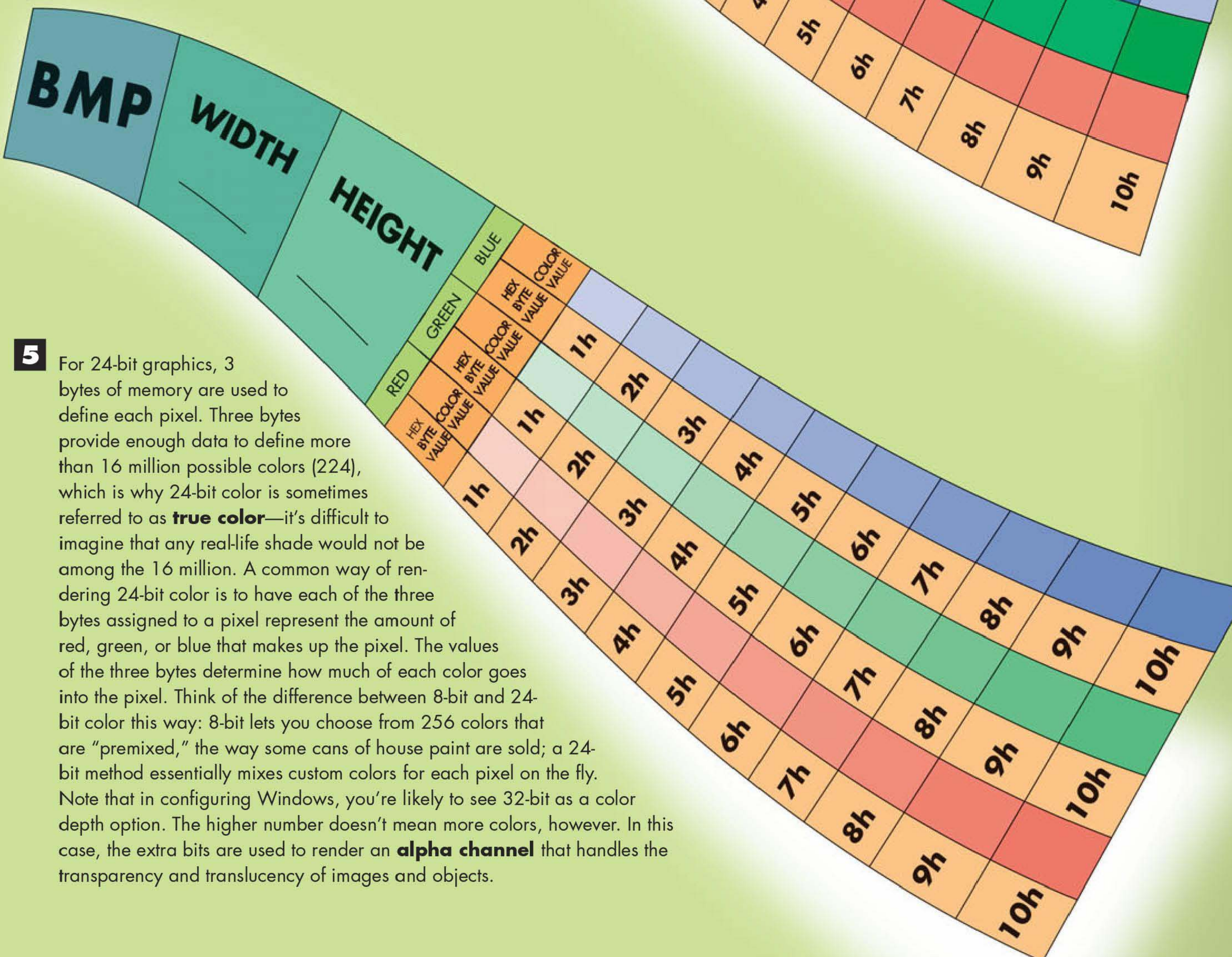




The last 7 bits are discarded because they fall outside the matrix of pixels set up in the header.

**3** In the memory set aside for video display, the bytes that make up the black-and-white image consist of some bits set to 1 and the rest to 0. A 1 means a pixel that corresponds to that bit should be turned on. A 0 bit indicates a pixel should be turned off. The man in the hat consists of 121 pixels, which in a black-and-white image can be stored in 16 bytes.

**4** A color bitmap requires more than 1 bit of information for each pixel. Eight bits (or 1 byte) per pixel are enough data to define a palette of 256 colors because 8 bits of binary information has a total of 28 possible values, or 256. (The values shown here are in *hexadecimal*, a BASE-16 number system that uses A–F as well as 0–9.) Each possible 8-bit value is matched in the palette to a specific combination of red, blue, and green dots that make up a single pixel. Although the dots of color are separate, they're close enough to each other that the eye sees them as a single point of blended color, called a **virtual pixel**.

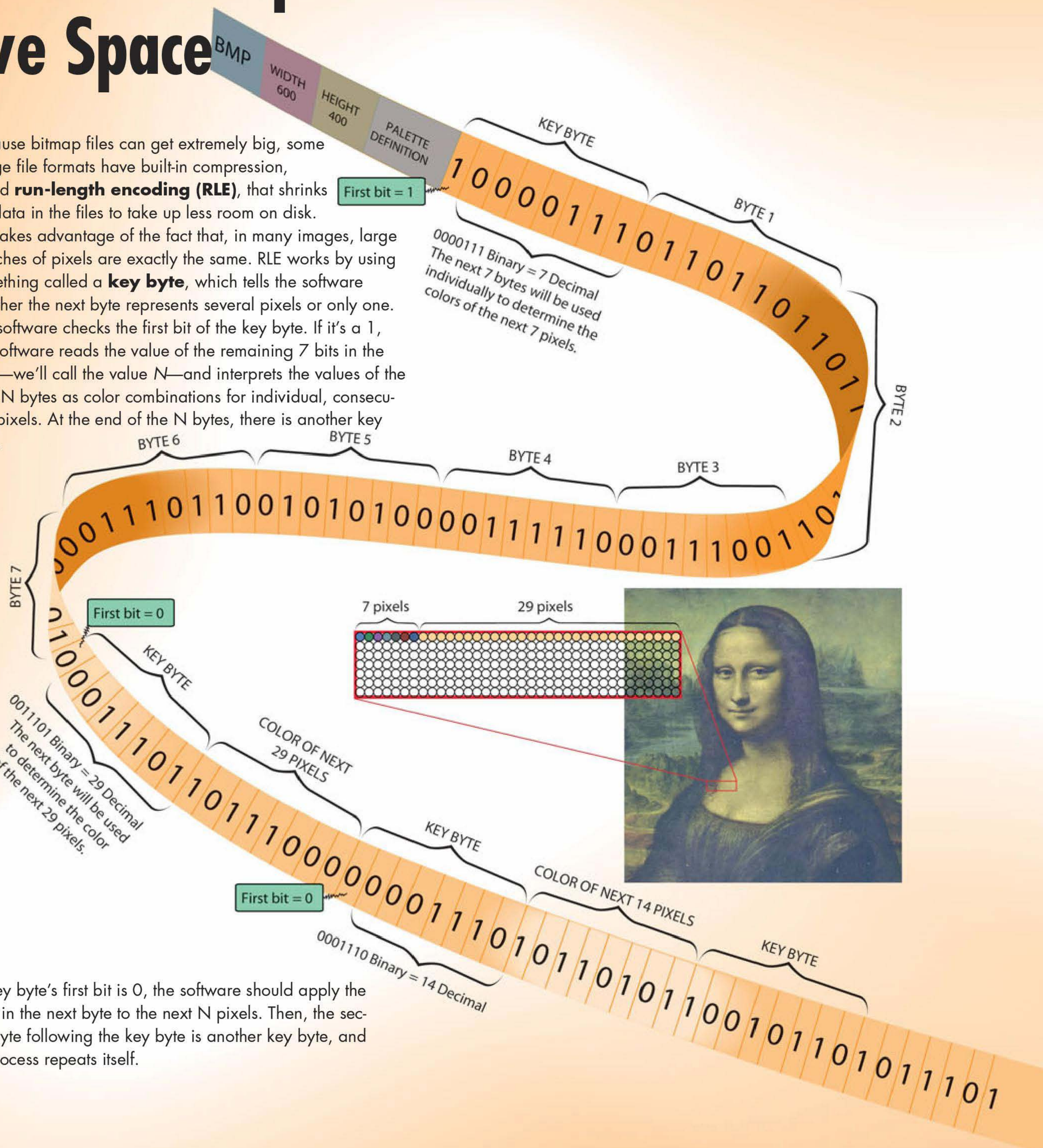


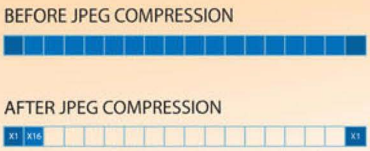

**5** For 24-bit graphics, 3 bytes of memory are used to define each pixel. Three bytes provide enough data to define more than 16 million possible colors (224), which is why 24-bit color is sometimes referred to as **true color**—it's difficult to imagine that any real-life shade would not be among the 16 million. A common way of rendering 24-bit color is to have each of the three bytes assigned to a pixel represent the amount of red, green, or blue that makes up the pixel. The values of the three bytes determine how much of each color goes into the pixel. Think of the difference between 8-bit and 24-bit color this way: 8-bit lets you choose from 256 colors that are "premixed," the way some cans of house paint are sold; a 24-bit method essentially mixes custom colors for each pixel on the fly. Note that in configuring Windows, you're likely to see 32-bit as a color depth option. The higher number doesn't mean more colors, however. In this case, the extra bits are used to render an **alpha channel** that handles the transparency and translucency of images and objects.

# How Art is Squeezed to Save Space

**1** Because bitmap files can get extremely big, some image file formats have built-in compression, called **run-length encoding (RLE)**, that shrinks the data in the files to take up less room on disk. RLE takes advantage of the fact that, in many images, large stretches of pixels are exactly the same. RLE works by using something called a **key byte**, which tells the software whether the next byte represents several pixels or only one. The software checks the first bit of the key byte. If it's a 1, the software reads the value of the remaining 7 bits in the byte—we'll call the value *N*—and interprets the values of the next N bytes as color combinations for individual, consecutive pixels. At the end of the N bytes, there is another key byte.



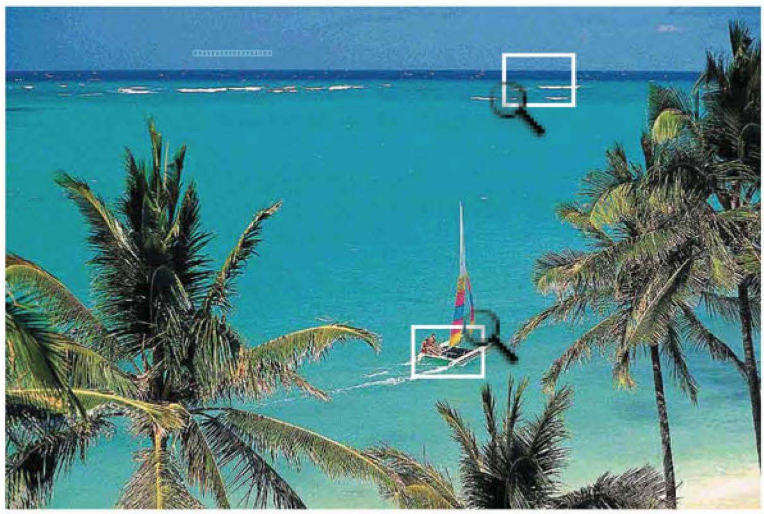

**2** If a key byte's first bit is 0, the software should apply the value in the next byte to the next N pixels. Then, the second byte following the key byte is another key byte, and the process repeats itself.

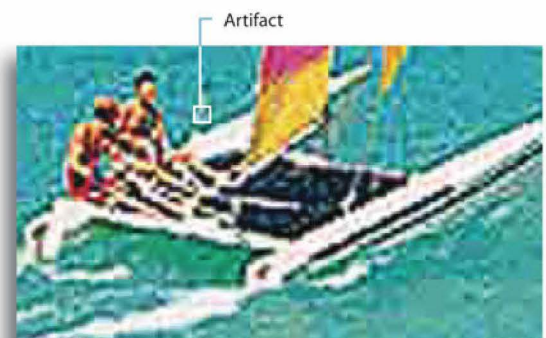



**3** Software, in a camera or computer, that compresses a file in the JPEG format examines the file for pixels whose absence is unlikely to be noticed by the uncritical human eye. More accurately, it's not their absence that's overlooked; it's the fact that they've been disguised to look like neighboring pixels. If half of the area of a landscape photograph is devoted to a cloudless sky, saving the color values of each of the pixels making up the sky is an extravagance when many of the pixels are exactly the same shade and intensity of blue.

**4** Rather than recording the 24 bits needed to describe each pixel, the compression software records the bits for only one—the **reference pixel**—and then writes a list of the locations of every pixel that is the same color.

**5** For more drastic compression, another trick is to divide an image into 8-by-8 pixel blocks and calculate the average color among the 64 pixels in that block. If none of the pixels are too different from the average, the compression changes the colors of all the pixels to the average; your eye is none the wiser.

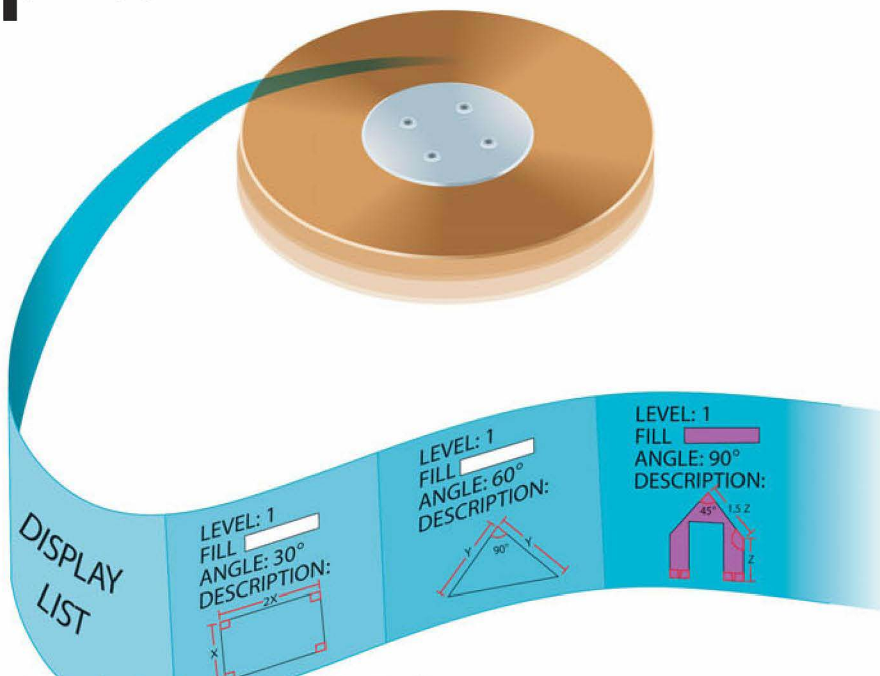


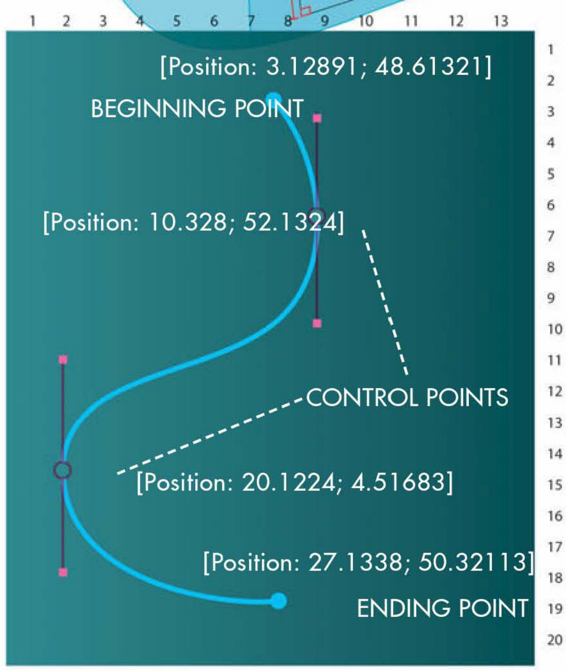

**6** A problem with JPEG compression has been that a photo is recompressed—with more original pixels thrown away—each time it is re-saved to the JPEG format. If it's saved too many times, you increase the appearance of unnatural **banding**—obvious strips of color instead of smooth gradations as you see, for example, in the sky. It also causes **artifacts**—shapes and colors that weren't in the original photo and are solely the product of mathematics. Eventually the photo may come to look **posterized**, so named after posters that have been simplified by using only a few colors.

**7** A more recent form of JPEG, called **JPEG 2000**—using a **.jp2** or **.j2k** file extension—compresses photos by using **wavelet** technology, a type of mathematical calculation that transforms pixel values into strings of numerical **tokens** for frequently recurring combinations of pixels, such as combinations of red and blue pixels that might be used to create a brown tree trunk. The strings replace the cruder blocks of JPEG compression and result in smaller files without the same types of visual artifacts seen in standard JPEG photos.

# How Vector Graphics Change Shape

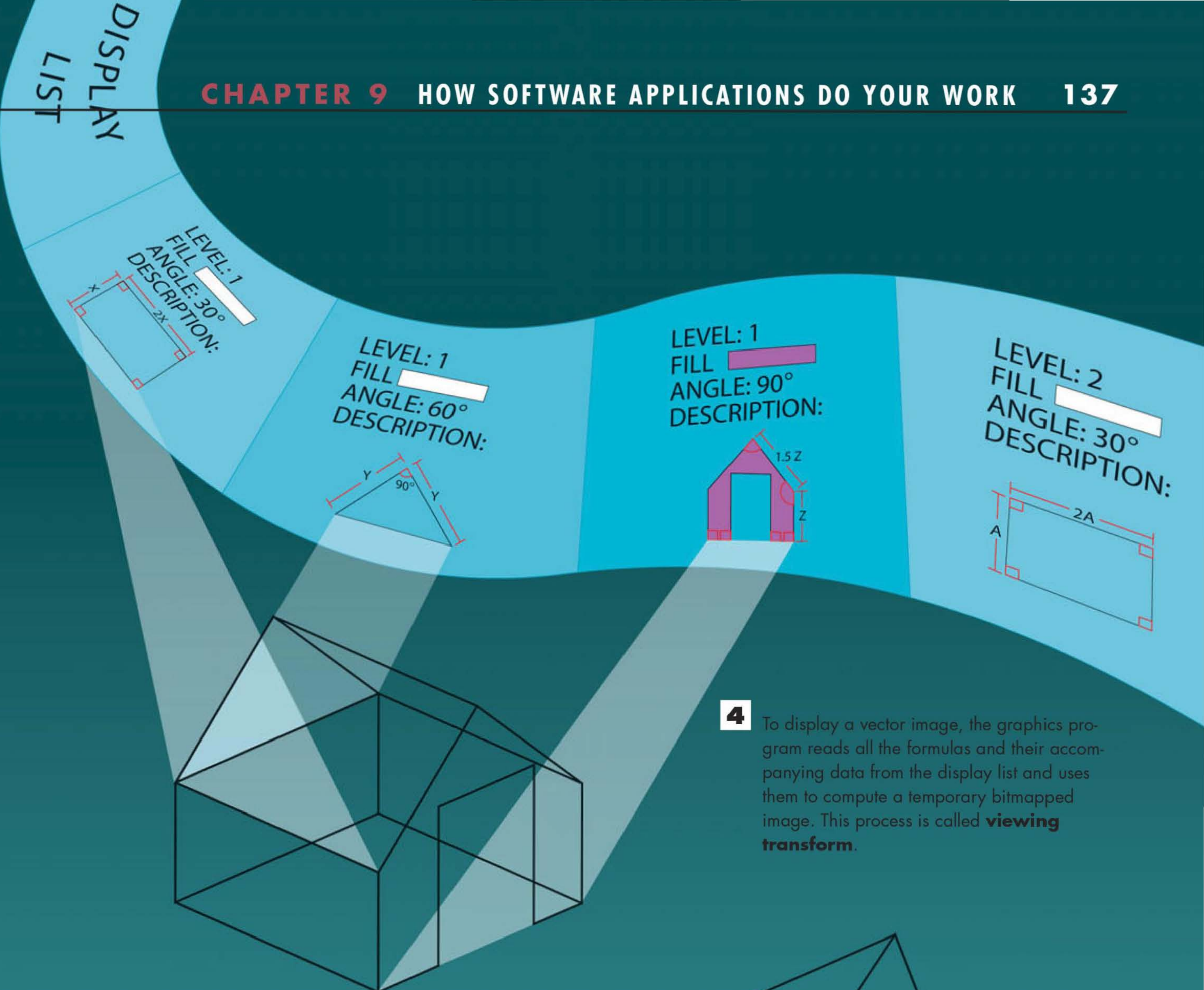



**1** A **vector-based graphic** image is stored in a file as a display list that describes in mathematical terms every shape, or object, in the image along with its location and properties, such as line width and *fill* (the color pattern that fills a shape). The display list also specifies the hierarchy of objects—which ones should be drawn first and which are on top of others. Vector graphics differ from bitmapped graphics, which are locked into an unchanging size and shape. Vector graphics can change size and shape by changing the math that defines them.



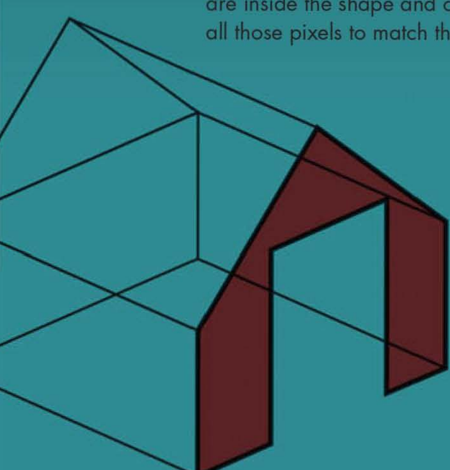

**2** To draw an object, the program must know the locations of only a few points. The formula for a Bézier curve, for example, needs only four points: the beginning point, the ending point, and two control points that determine how far the curve is "pulled" away from a straight line.

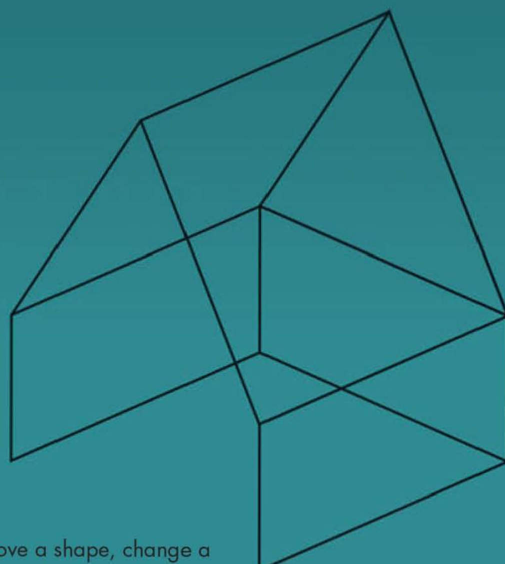
**3** The points are each defined by two numbers—one for the point's vertical position and the other for the horizontal position. Each of those numbers is stored to a much higher degree of precision than the pixel locations alone could specify. This precision allows the software to draw the curve accurately, no matter how much its size is increased or decreased, or if it's moved or otherwise manipulated.




**4** To display a vector image, the graphics program reads all the formulas and their accompanying data from the display list and uses them to compute a temporary bitmapped image. This process is called **viewing transform**.

**5** To display a **fill**, the program computes a mathematical formula that determines the locations of all the pixels that make up the edge of the shape. With that information, the program can determine which pixels are inside the shape and change the color values for all those pixels to match the fill.

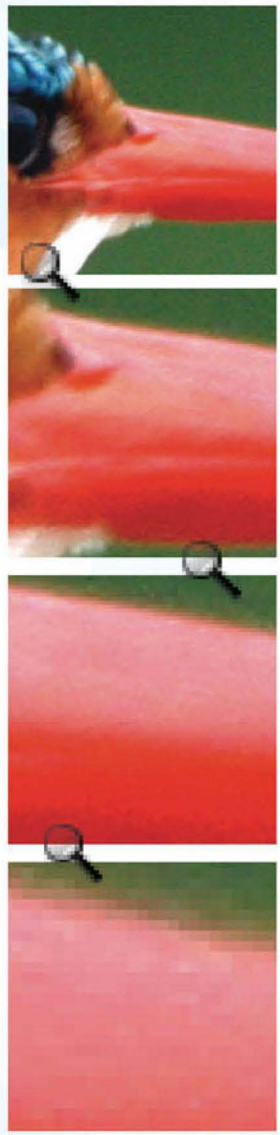




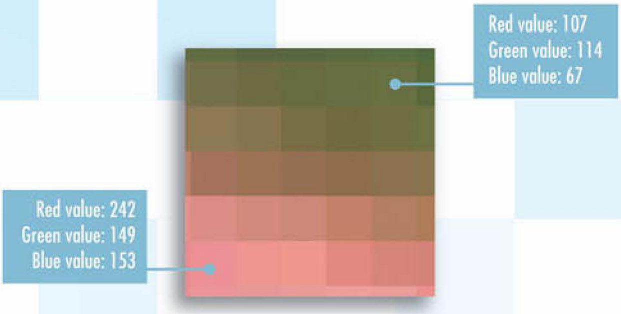
**6** Any time you change or move a shape, change a shape's attributes (such as its color), or add a new object to the image, the software changes the data stored in the display list for all the affected objects. Data for any object not changed is left unmodified. Viewing transform then recomputes the display bitmap to update the screen.

# How Software Uses Numbers to Correct Photos

The humongous advantage of using numbers to represent images is that you can lighten or darken an image, bring out contrast, sharpen or blur it, turn it upside down, or transform it into a psychedelic abstract, all by using simple arithmetic. It's not so simple that we can sit at a computer with the Mona Lisa on the screen and subtract numbers from the color values in Mona's portrait, turning her smile into a frown. It would be an excruciating job, the kind of job computers are made for. A PC can calculate a frown job on Mona in less than 10 seconds. It's all just a numbers game.

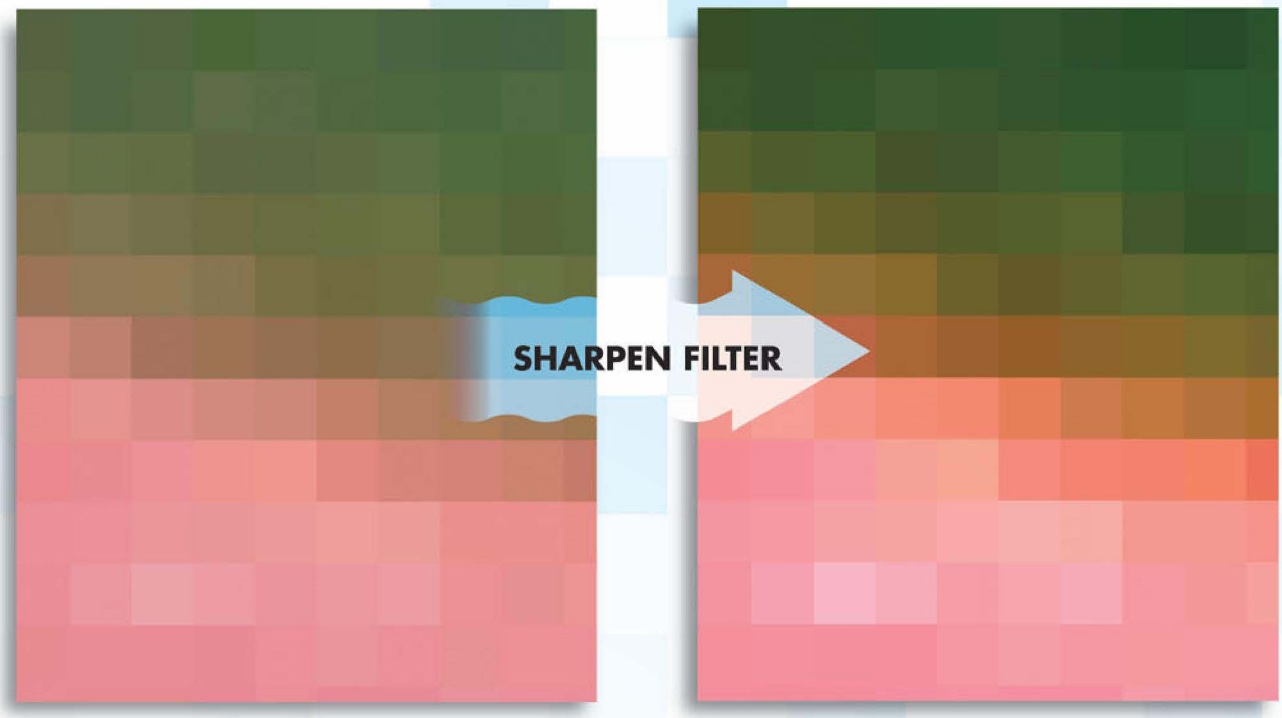


1 Your digital camera records your digital photo as a set of tiny pixels. As you look closer at a photo, you begin to see the pixels as jagged edges and then squares. What you thought were distinct lines and boundaries turn out to be more gradual transitions between colors.




**2** Zoom in far enough and you see that each pixel is actually a tiny square consisting of a single, solid color.

**3** When you tell your image editing software to make a change, such as sharpen, brighten, or apply effects, the software looks at the differences in color between pixels. Often, it compares those values with pixels that are close but not touching the pixels under scrutiny.



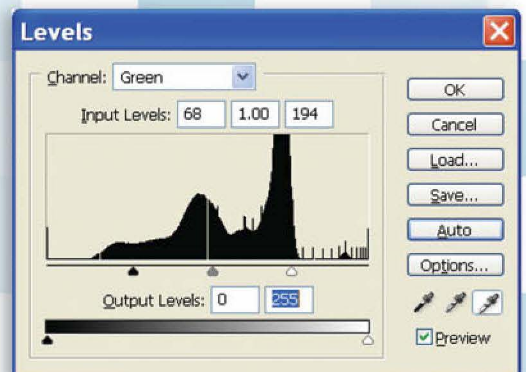

**4** The software applies a mathematical formula to recalculate the color of each pixel. It starts with the formula for the type of editing or effect you are asking it to do. It also takes into account the differences in color between pixels, often those in or near transition areas.

**5** Not all pixels are changed equally. Some pixels might retain the same general color but take on a slightly darker or lighter tone. Other pixels might be assigned a very different color. Pixels that are not near a border area might not be changed at all.

# How Photo Editors Restore Old Pictures

Color photos are subject to the same slings and arrows of outrageous shoe boxes that torment black-and-white pictures—the dirt, the creases, the spills, the humidity, and of course the fatal reactions that occur when the still-active chemicals of one photograph are crammed up against those of another picture for a decade or so. But the chemicals in color photos are still more volatile and more susceptible to turning the only picture of beloved Uncle Ernie into what looks like an extraterrestrial blob of protoplasm. Luckily for all the myriad misfortunes that can affect photos, we have an arsenal of E.R. weaponry to bring them back from the brink. There are so many tools in the digital darkroom, there's no space to cover them all in detail, but here are some of the most common devices available in programs such as Photoshop, Elements, and Paint Shop Pro to bring color—the right color—back to the cheeks of fading ancestors.

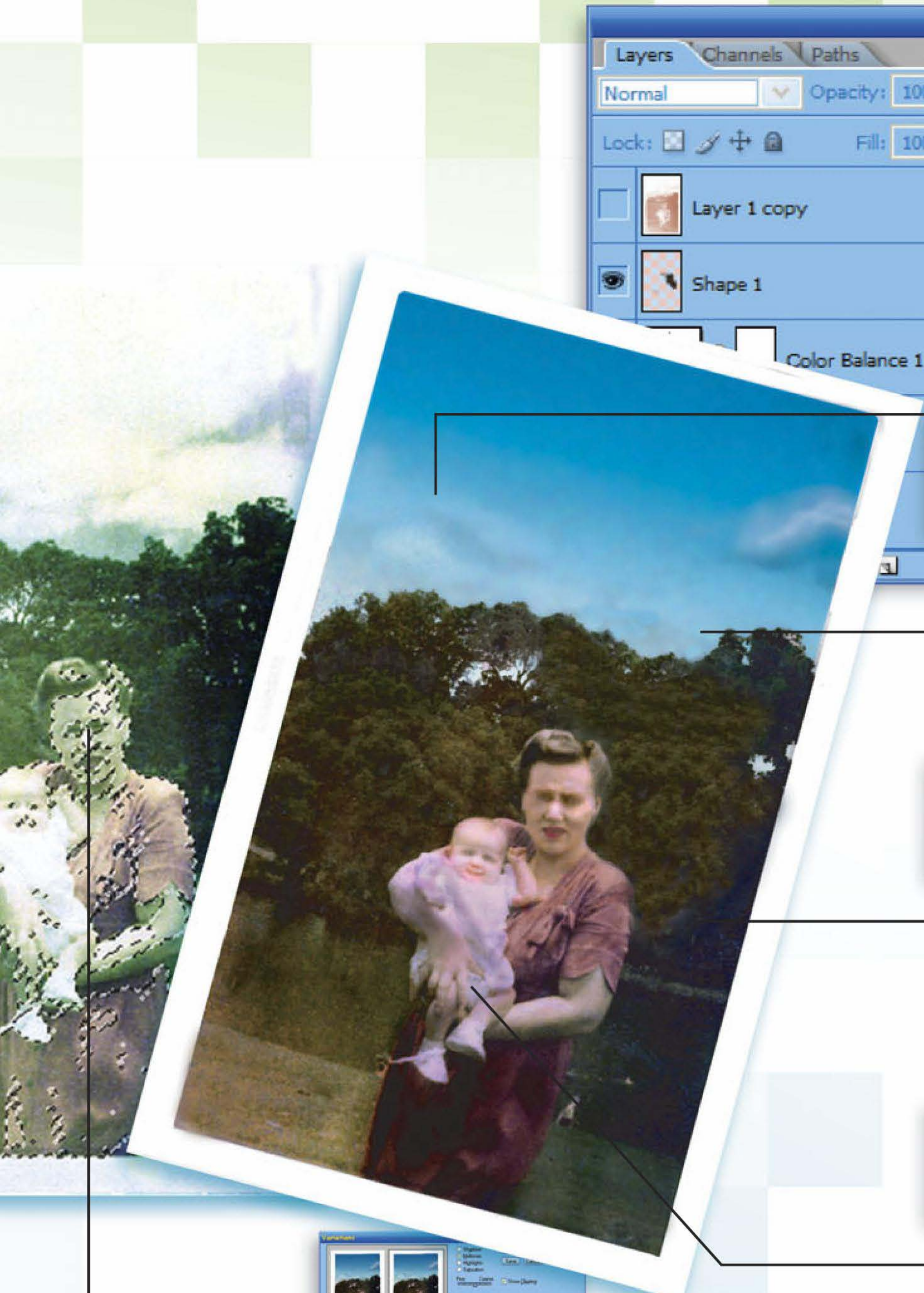
**Levels** (histograms) is the most versatile of several methods to isolate and tame discolorations that have taken over the photograph.




**Burn** and **Dodge** duplicate the techniques of the old chemical darkroom so the retoucher can darken and lighten specific portions of the photo.

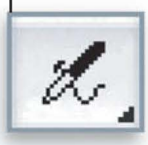
**Healing brush** repairs numerous dust spots, scratches, and assorted and unexplained flaws by duplicating the pixels surrounding the damage so they cover the defects seamlessly.

**Selection tool** isolates the mother and baby so they can be worked on without affecting the background. The selection can also be reversed to work on the background without worrying that touch-ups will spill over onto the mother and child.




**Layers** provide a way of working on a duplicate of the photograph within the same file and then controlling how changes are blended into the original picture. Two duplicate layers of the washed-out image that emerged from color changes were *multiplied* to increase the contrast and color depth of the photo.

**Gradient tool** fills a sky that has lost all hint of color, permitting the blue to fade as it approaches the horizon.



**Airbrush** brings out clouds that have been submerged by the blue gradient. The airbrush also adds a hint of eyeballs that have been lost entirely in the shadows of the mother's eyes.

**Cloning tool** covers bigger and more complex flaws by copying, through a sort of artistic wormhole, good portions of the photo to replace flawed areas with the same control you have using a brush. Here, some of the dark trees on the right were replaced with light trees from the left of the photo.

**Variations** are used as a final touch so the retoucher is able to see and choose from a selection of thumbnail versions of the same photograph in which hues and brightness are slightly varied. This enables the retoucher to see, at a glance, which variation produces the most pleasing result.

**Sharpen tool** restores definition to edges that have become blurred through fading or by the retouching itself.

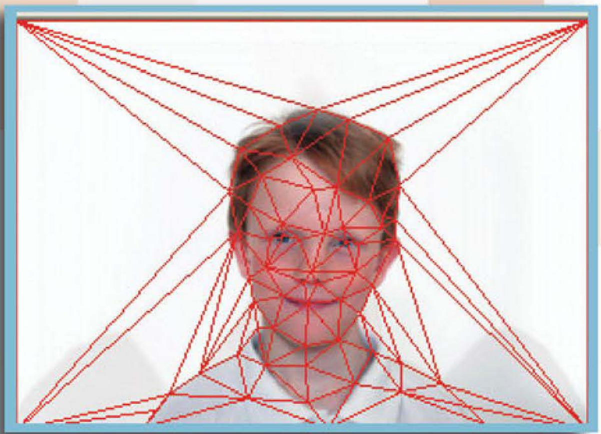
# How the Digital Darkroom Pokes Fun at You

Raising a young boy to be a man is no easy chore—unless you have morphing software. **Morphing** converts one image into another in a number of discrete steps, each of which is a blend of the characteristics of both. Although there are many mathematical methods of morphing, the basic procedure is a combination of two techniques: cross dissolving and warping.

1 **Cross dissolving** is the same technique used in slideshows and PowerPoint presentations: One image fades out while another fades in. A computer producing the effect begins by holding both images in memory simultaneously as it displays the first image.

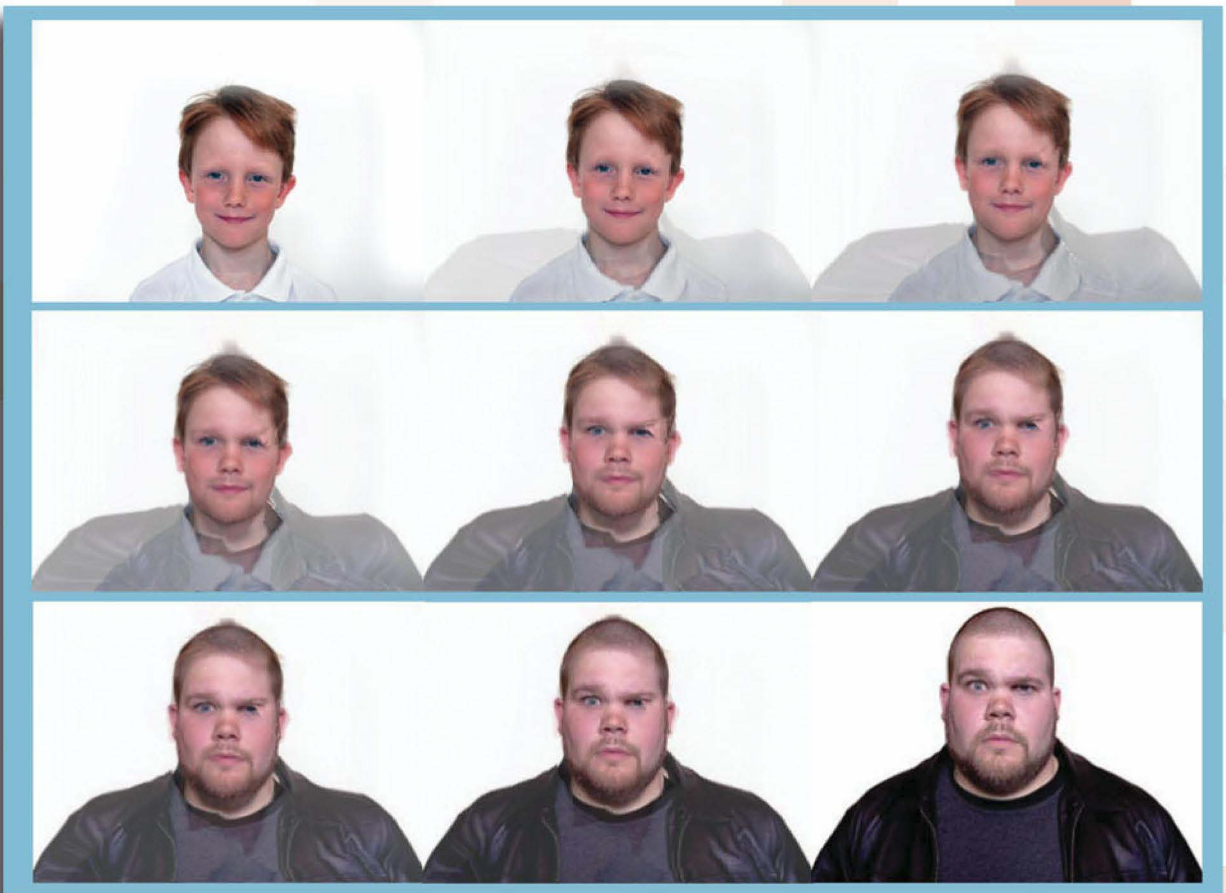
2 The morphing software randomly chooses some of the pixels in the first image and replaces them with the pixels from the same positions in the second photo. It repeats the process until it replaces all the original pixels. This gives us a dissolve, a commonplace feature of any movie. But it's not yet a morph. It needs to warp as well.

3 In one common **warping** method, the user manually places dots on one face and partner dots on the other face to identify major features, such as the nose, mouth, and eyes. The more pairs of dots used in the warp, the more realistic the result. The program here uses a variety of colors for the dots, to help the user keep track of what's been done.



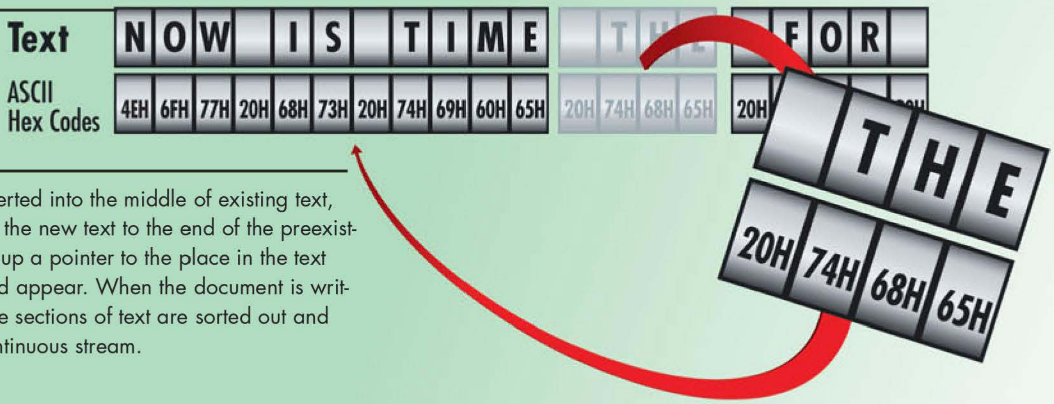
**4** The software also makes a record of what is inside each of the regions the dots define. It uses this information when it performs a cross dissolve, replacing pixels in one photograph with pixels from the other. The software simultaneously repositions the transported pixels and the polygons they form to warp the emerging image so the pixels move steadily toward the positions they occupied in the picture from which they came.

**5** The end result is the same as the cross dissolve: One picture replaces the other. But when all the stages are displayed in an animation, warping makes the subject of one photo morph gradually into the other. (By working with two copies of the same photo, you can warp without dissolving. The effect is to distort the original image so it grows a bigger nose, longer hair, or bulging eyes.)



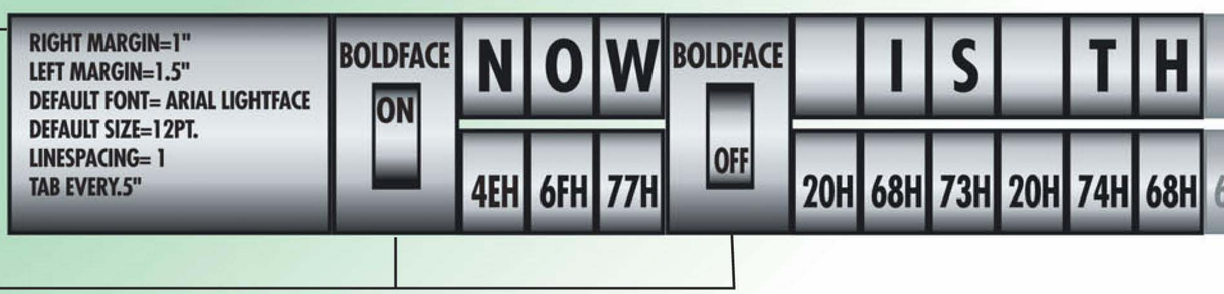
# How Word Processors Format Text

**1** When you type characters into the word processor, they are stored as ASCII codes, shown here as numbers, in a section of memory set aside for that document.




**2** If new characters are inserted into the middle of existing text, the word processor adds the new text to the end of the preexisting text in RAM and sets up a pointer to the place in the text where the new text should appear. When the document is written to a disk, the separate sections of text are sorted out and written to the file in a continuous stream.

**3** A special section of the document, called the **header**, stores information about the file, such as the default font, margin settings, tab settings, and other data that is applied throughout the document unless a specific change is made.



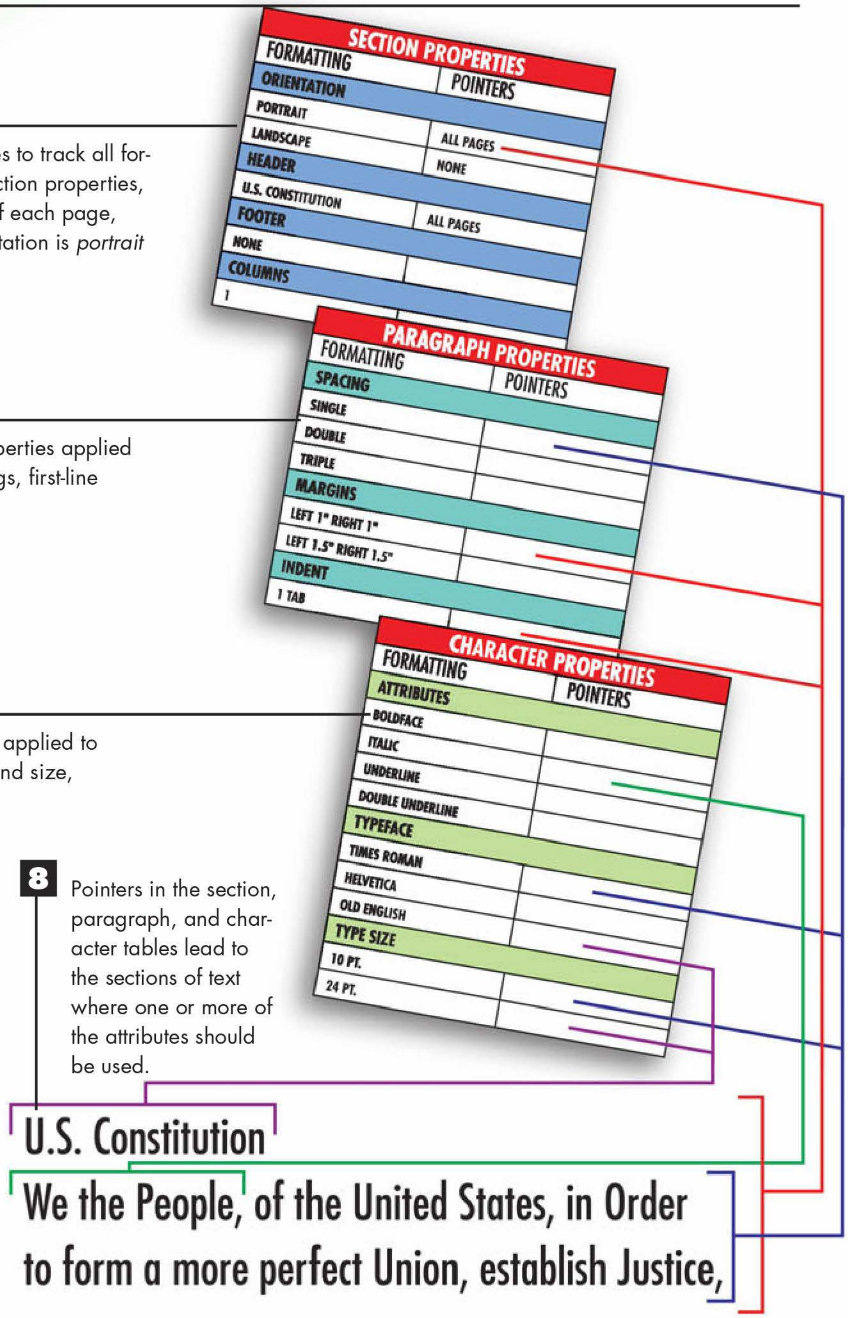

**4** When you change a section of text by applying an **attribute**, or **formatting**—such as boldfacing, italics, underline, margin settings, and type size—the word processor uses one of two methods to track format changes. WordPerfect for Windows and HTML, the formatting language used to create Web pages, uses inline formatting. **Inline formatting** inserts its own code for that attribute where it's supposed to start. At the end of the affected section, the formatting software inserts another code that signals the end of the formatting. These codes are not usually displayed onscreen.

**5** Microsoft Word uses a collection of tables to track all formatting information. One table tracks section properties, such as headers that appear at the top of each page, tab settings, and whether the page orientation is *portrait* (vertical) or *landscape* (horizontal).

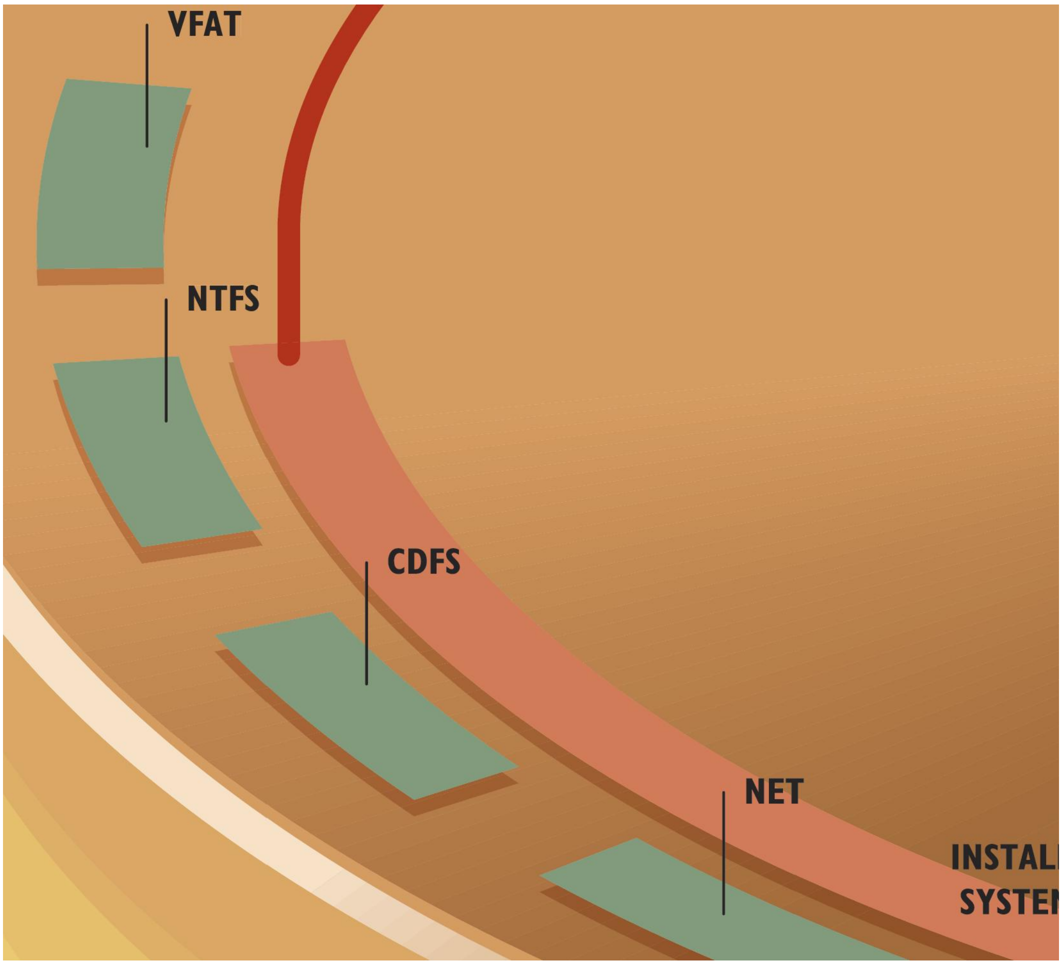
**6** A second table tracks the formatting properties applied to paragraphs, such as the margin settings, first-line indentations, and line spacing.

**7** A third table tracks formatting properties applied to individual characters, such as typeface and size, boldfacing, italicizing, and underlining.

**8** Pointers in the section, paragraph, and character tables lead to the sections of text where one or more of the attributes should be used.




**9** As you type or scroll through a document, the word processor reads more text and formatting codes from RAM and sends its own commands to Windows, which passes them to the **display driver**, a collection of codes for controlling your specific display adapter card. Finally, the card sends to the monitor the electrical signals that turn pixels on and off as needed to display text, graphics, or any other part of the document. When the word processor sends text to be printed, a printer driver performs a similar function to translate text and formatting into the patterns of dots created by all printers, whether dot matrix, ink-jet, or laser.


**30,000 B.C.** Paleolithic peoples in central Europe record numbers by notching up tallies on animal bones, ivory, and stone.

**1640** First punch cards for storing data are invented by Jacquard. Punch cards are used by the first electronic computers in the 1940s and onward until the development of more reliable data storage.

**1857** Sir Charles Wheatstone uses paper tape to store data. This technique for data storage is similar to punch cards except the tape can be fed continually through the machine.

**1890** Herman Hollerith uses cards to store data information, which are fed into a machine that compiles census results mechanically. As many as 80 variables can be stored on a single card. Instead of 10 years, census takers compile their results in just six weeks with Hollerith's machine.

**1940** Vacuum tubes are used for storage for the next 10 years.

**1944** First stored program computer is invented, the EDVAC.

**1949** The basic concept for core memory is patented by An Wang of Harvard University, but his technique involves using the cores on single wires to form delay lines.

**1950** Tape drives start to replace punch cards.

**1951** UNIVAC is delivered to the U.S. Census Bureau three years late. It's a hit, with revolutionary features such as mercury delay lines for memory and magnetic tape for input instead of punched paper.

**1952** Magnetic drums appear on the scene.

# PART 4

# Data Storage

## CHAPTERS



**1953**
Jay Forrester and a team at the Massachusetts Institute of Technology install magnetic core memory into the Whirlwind computer, giving it a twice-as-fast access time of six microseconds.

**1954**
The Burroughs B205 uses vacuum tubes and a magnetic drum main memory system. An arithmetic operation takes several milliseconds.

**1956**
IBM introduces the 305 RAMAC (random access method for accounting and control), the first magnetic hard disk storage system. The RAMAC stores 5 megabytes (MB) of data, is the size of two large refrigerators, and costs $10,000 per MB. The device can store five million characters of data on 50 disks, each 24 inches in diameter. Each disk can hold the equivalent of 25,000 punch cards.

**1957**
The first hard drive is introduced as a component of IBM's RAMAC 350. It requires 50 24-inch disks to store 5MB of data and costs roughly $35,000 a year to lease, or $7,000 per megabyte per year.

**1962**
Teletype ships its Model 33 keyboard and punched-tape terminal, used for input and output on many early microcomputers.

**1963**
Control Data Corporation announces and delivers the 3600 computer, 603 tape drive, and 405 card reader.

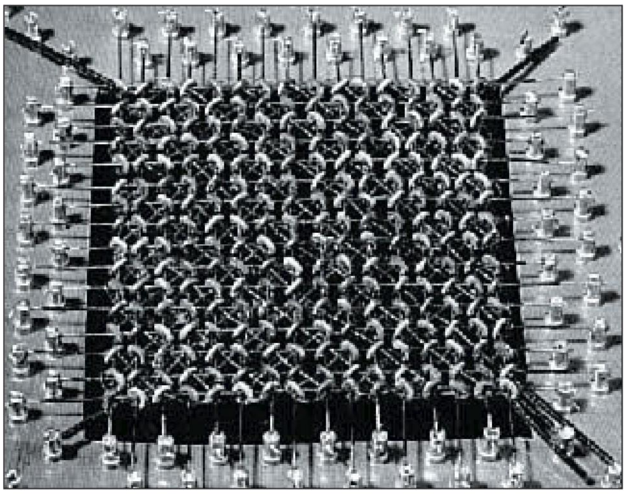
**1967**
IBM builds the first floppy disk.

**1969**
Intel announces a 1KB RAM chip, which has a significantly larger capacity than any previously produced memory chip.

*The beauty of mechanical problems is that they are often visible to the naked and untrained eye. If white smoke is rising from a disk drive, that is probably where the problem lies (unless your disk drive has just elected the new Pope).*

—John Bear, *Computer Wimp*

**THE** problem that always faces computer engineers is this: When the programs, data, and calculations that computers use aren't needed for a while, where do you keep them? Through the years, the answer to that question has changed as faster and more capacious methods of mass storage have been created. In the mid-1800s, holes punched into stiff paper cards were used to feed data and programming to early calculators and other machines.



This mesh of wires and iron donuts held 256 bits of memory. Today's memory chips could hold more than a billion bits in the same space.

Data storage didn't improve much until the early 1950s, when magnetic tape replaced punch cards for **offline storage**, as opposed to **online storage**, which is always available at any time to a computer. There was a serious problem with magnetic tape. Data had to be written to it and read from it in sequence. Some reels of tape were more than a foot in diameter and contained thousands of feet of tape. If the spot on the tape you needed to get to was at the end of a reel of tape, you had to travel the entire length of tape before you could read or write your data. That meant data and programs had to be read completely, and be small enough to be held in a computer's memory while the program crunched the data. Within the computer's core memory, the data could be accessed *randomly*; the computer could go directly to the data it needed, along the way skipping over any other data in **memory**. **Core memory**, used extensively in the 50s and 60s, was made of ferrite donut-like cores, each about 1/16th of an inch in diameter, strung on a grid of wires. Sending current through the wires caused the cores to hold a charge, to represent 0 or 1. The storage was permanent; the computer retained it even if the power was turned off. But core memory took up too much space to be practical as a mass storage medium.

**1971**
Intel introduces the 1101 chip, a 256-bit programmable memory, and the 1701 chip, a 256-byte erasable read-only memory (EROM).

**1972**
The Altair, the first "personal computer," comes out, with 256 bytes of memory (expandable to 64K).

**1973**
IBM releases the 3340, the first Winchester hard disk. It has a capacity of 70MB spread over four platters—or enough for more than 35 complete transcripts of the Watergate special prosecutor's 60 hours of audio tapes. The recording head rides on a layer of air 18 millionths of an inch thick.

**1978**
Apple Computer introduces the Disk II, a 5.25-inch floppy disk drive linked to the Apple II by cable. Price: $495, including controller card.

**1971**
IBM introduces the "memory disk", or "floppy disk." It is the industry's first flexible magnetic diskette, an 8-inch floppy plastic disk coated with iron oxide, ushering in the era of data portability and desktop computing. The floppy disk greatly increases the convenience of data handling and becomes widely used as a basic storage medium for small systems.

**1976**
iCOM advertises its "Frugal Floppy" in *BYTE* magazine, an 8-inch floppy drive selling for $1,200.

**1976**
Shugart announces its 5.25-inch "mini-floppy" disk drive for $390.

**1978**
Dynacomp starts business and distributes software on paper tape.


In the early 80s, this was the leading edge of PC storage—two 5-inch floppy drives were bigger and heavier than many of today's laptop computers.

The computers then did not have the convenience of using disk drives as temporary scratch pads while they reordered information or calculated new results. Whenever a job called for new data, a computer technician would have to find the right tape among many on a shelf, load it onto the tape drive, and run the tape until the computer found the data it needed.

The first **hard drive** appeared in 1957 as part of IBM's RAMAC 350. Reynold B. Johnson, an IBM engineer and a prolific inventor, had to overcome the doubts of his fellow scientists in coming up with a model for the RAMAC Disk File. It required 50 24-inch disks to store five megabytes—the same amount we can easily store today on four floppies. It weighed one ton and cost roughly $35,000 a year to lease, or $7,000 a megabyte per year.

For years, "disk farms" of 14-to 18-inch hard drives, housed in cases the size of washing machines, were expensive components of large business systems. The first IBM PC, released in 1981, had no hard drive, only one or two drives that held a couple of hundred kilobytes of data or programs on floppy disks 5.25 inches square. It also included a port for saving data to a tape drive, possibly the most underused component in PC history. The IBM-XT, released in 1983, had a hard drive twice as thick as today's notebook PCs and held 5-10MB of storage.

**1980**
Seagate makes the first 5.25-inch hard disk. It has a capacity of 5MB spread over four platters, equivalent to almost 1,300 cartridges of Atari's *Pac-Man* game. Price: $600 ($120 per megabyte). It fits in the same space as the floppy disk drive.

**1981**
IBM introduces "thin-film" head technology, which enables the 3380 hard disk drive to read and write data at three million characters per second; it is the first commercial unit to achieve such a rate. The thin-film read-write head of an IBM 3380 disk file flies 12-millionths of an inch over the disk surface, comparable to a large plane flying 1/20th of an inch over a lake's surface without touching the water. With a storage capacity of 12 million bits per square inch, the 3380 offers 6,000 times more storage per square inch than the original RAMAC disk drive.

**1980**
Sony Electronics introduces the 3.5-inch floppy disk drive, which is double-sided, double-density, and holds up to 875KB unformatted.

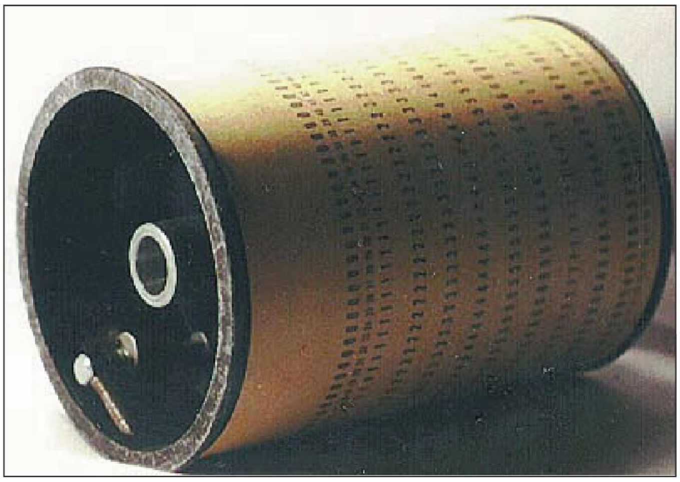
**1981**
Microsoft's new DOS (Disk Operating System) requires less than 160K of disk space.

**1981**
"640K ought to be enough for anybody."—Bill Gates

**1982**
The compact disc comes out.

**1982**
Iomega begins production of the 10, a 10MB 8-inch floppy-disk drive using Bernoulli technology.

**1982**
Drivetec announces the Drivetec 320 Superminifloppy, offering 3.33MB unformatted capacity on a 5.25-inch drive.



A metal cylinder studded with punches was used until the 80s to store data in the form of hole punches into heavy paper cards.

The term "hard" came from the fact that the disks were made of a rigid aluminum alloy, in contrast with the literally floppy mylar disk used in floppy drives. (The inner disk of today's smaller, 3.5-inch floppies are also made of mylar, but they're housed in a rigid plastic case, and improved disk coatings and devices for recording and reading data have raised capacity to 2.44MB on conventional floppies and as high as 2.5GB on floppies such as the Zip drive.) The disk material was not the only difference. The magnetic coating on the disk, or **platter**, was denser so that more data could be recorded in a smaller area. The **read/write head**—a combination of electromagnets and sensors—did not touch the hard disks. This avoided the wear and tear in floppy drives caused by the read/write heads actually touching the disk surface. It also allowed the rigid disk to spin much faster so that data could be accessed rapidly. It also created the phenomenon of the hard disk **crash**, a disastrous malfunction in which the read/write head literally crashes into the platter, carving a small furrow through the coating.

Since 1983, the size and cost of hard drives have shrunk while the capacity has grown. By the mid-1980s, a standard drive measured about three inches high and weighed only a few pounds. By 1987, the 3.5-inch form factor began to appear. These compact hard drives weigh as little as a pound and are about the size of a paperback book. For a few hundred dollars, it's now possible to buy a 750GB drive that fits into the same space as a small floppy drive. No individual technology breakthroughs are responsible for this fortunate situation. Instead, hard drives have evolved, one component at a time, from motors to disk coating to smaller and more sensitive read/write heads. Such slipstream improvement is typical of all computer components.

**1982**
Amdek releases the Amdisk-3 Micro-Floppy-disk Cartridge system. It houses two 3-inch floppy drives designed by Hitachi/Matsushita/Maxell. Price is $800 without a controller card.

**1982**
Tabor demonstrates a 3.25-inch floppy disk drive, the Model TC500 Drivette. Unformatted capacity is up to 500KB on a single side.

**1982**
At the West Coast Computer Faire, Davong Systems introduces its 5MB Winchester Disk Drive for the IBM PC, priced at $2,000.

**1983**
Sony Electronics announces the 3.5-inch floppy disk and drive, double-sided, double-density, holding up to 1MB.

**1983**
With the introduction of the IBM PC/XT, hard disk drives become a standard component of most personal computers.

**1983**
Philips and Sony develop the CDROM as an extension of audio CD technology.

**1984**
Random access memory (RAM) becomes available.

**1984**
Apple Computer introduces the DuoDisk dual 5.25-inch floppy disk drive unit for the Apple II line.

**1985**
The first CD-ROM drives make their debut on PCs, featuring a 650MB read-only capacity—enough for 74 minutes of digital audio.

**1985**
Apple Computer introduces the UniDisk 5.25 single 5.25-inch floppy disk drive, with the capability to daisy-chain additional drives through it.

Not that new technologies haven't expanded the capabilities of drive storage. The use of the laser in multimedia disks has vastly increased the amount of data that can be stored on the same surface area. But on a cost per megabyte, magnetic storage is still the best bargain, and optical technology is still too slow to be a workhorse everyday drive. We'll look at all these technologies in this part and the book's section on multimedia. And we'll even look at tape storage, which after all these years and all these changes, is still being used.

## KEY CONCEPTS

**access time** The time, measured in milliseconds, from when the access command is given to when the read/write head is positioned to read or write a specific sector. The access time for hard drives is a combination of seek time, controller overhead, and rotational latency.

**actuator** The mechanism that moves read/write heads to the correct cylinder. A rotary voice coil and head gimbal assembly control movement. The actuator houses the heads at the tips of its arms.

**Advanced Technology Attachment (ATA)** The command protocol used with the parallel ATA and serial ATA interconnects.

**Advanced Technology Attachment Packet Interface (ATAPI)** An extension to ATA for controlling optical (CD and DVD) drives.

**average latency** Derived from the spindle speed, this is the average amount of time it takes for the drive to rotate to the correct address, so that the head can begin reading or writing in its desired location.

**bad block/sector** The area on a drive platter that is damaged and not reliable. The locations of such sectors are recorded in a special area of the drive so the drive does not attempt to write data to them.

**buffer** Part of a drive that temporarily stores data in memory chips. The buffer compensates for differences between the time for transferring data and the time it takes to process the data.

**CRC (cyclic redundancy check)** Part of a sector that detects missing data within a sector. It is used to test that the data retrieved from a disk is without error. When data is written to disk, a special check number (CRC) is calculated, based on the data itself, and stored with it. As data comes off the disk, the CRC is recalculated and compared with the stored CRC. The two match if no errors are present.

**command protocol** A set of commands that tell drives what actions to take, such as reading and writing data.

**cluster** One or more successive sectors that contain a continuous group of data. Also, the smallest unit in which data is stored on a drive.

**compression** A process to remove redundant data so that a file is smaller.

**cookie** The Mylar disk to which data is written in a floppy drive.

**cylinder** The vertical position of all a drive's read/write heads over their corresponding platter on a specific track, forming a vertical cylinder.

**1986**
5.25-inch form factor hard drives shrink considerably in terms of height. A standard hard drive measures about three inches high and weighs only a few pounds, while lower capacity "half-height" hard drives measure only 1.6 inches high.

**1987**
3.5-inch form factor hard drives begin to appear. These compact units weigh as little as a pound and are about the size of a paperback book. They are first integrated into desktop computers, and are later incorporated into laptops weighing less than 12 pounds.

**1988**
The PrairieTek 220 is released—the first 2.5-inch hard disk for portables. Capacity: 20MB spread over two platters, for a total of just over two volumes of Encyclopedia Britannica.

**1991**
Tandy introduces its low-cost CDR-1000 CD-ROM drive for PCs, including drive and controller card. It is about half the price of other drives.

**1991**
Insite Technology begins shipping its 21MB 3.5-inch floppy disk drive to system vendors. The drive uses "floptical" disks, using optical technology to store data.

**1992**
The cost of purchasing a 200MB hard disk drive drops below $200, or less than one dollar per megabyte.

**1992**
Crystal holograph memory is introduced.

**1992**
1.8-inch form factor hard drives appear, weighing only a few ounces and delivering capacities up to 40MB. A 1.3-inch hard drive, about the size of a matchbox, is introduced.

**1993**
NEC Technologies unveils the first triple-speed (450Kbps) CD-ROM drive.

**DMA (direct memory access)** A way to transfer data directly between the drive and RAM without the assistance of the CPU.

**data transfer rate** The number of bytes or megabytes transferred from a drive to memory (or from any storage device to another) in a second.

**directory** A set of related files; also called a folder in Windows. The files can be located physically on different parts of a drive, but they are grouped logically in a directory, which can contain other folders or subdirectories.

**drive** Any device for storing computer files.

**drive array** Two or more drives linked to improve file retrieval time and provide error correction.

**EIDE (enhanced integrated device electronics)** Enhanced IDE is a standard interface for mass storage drives. EIDE's enhancements to IDE (integrated drive electronics) make it possible to address a hard disk larger than 528MB, provide faster access to the hard drive, support for direct memory access (DMA), and support for additional drives, including optical drives and tape devices through the AT attachment packet interface (ATAPI).

**FAT (file allocation table)** A table the operating system maintains on a magnetic disk that provides a map of the clusters that contain each file.

**flying height** The height at which the head moves over a drive's platters. The lower the height, the greater the density of bits recorded to a specific area.

**form factor** Standard sizes and shapes of drives, such as 5.25 inches, 3.5 inches, and 2.5 inches.

**format** The process by which a disk is divided into tracks and sectors so that files can be stored and found in an orderly fashion. Also, the arrangement of a magnetic track pattern on the media that enables the drive to store data in an organized manner.

**fragmentation** The process by which files become broken up into widely separated clusters. Defragging or optimization corrects fragmentation by rewriting broken files to contiguous sectors.

**gigabyte** 1 Gigabyte = 1,073,741,824 digital bytes, commonly thought of as 1,000,000,000 bytes. PC hard drives today use hundreds of gigabytes to express a drive's capacity for data.

**hard error** A drive data error that cannot be overcome by repeated reading of the same spot on a drive's platter. A physical defect in the surface of the disk is usually the cause.

**head crash** A serious malfunction that happens when a read/write head accidentally touches the surface of a platter, damaging the drive and making data stored there unreadable and possibly making the entire drive unusable.

**head landing zone** An area on the media to which the read/write head returns when it is not active. No data is recorded in the zone so that if the head crashes into the platter there, no data is lost.

**hot plug** The ability to remove or add a drive (or other device) without having to turn a computer off.

**IDE (Integrated Drive Electronics)** A standard software for ATA (AT attachment) drives that allows the drive to work with any PC.

**1994**
NEC Technologies ships its quad-speed CD-ROM.

**1994**
Iomega Corp. introduces its Zip drive and Zip disks, floppy disk-sized removable storage in sizes of 25MB or 100MB.

**1995**
IBM introduces lightning-fast disks with new MR heads (Magneto-Resistive). Before this, the head technology was called thin film heads. Today, almost all vendors use MR heads.

**1997**
Quantum introduces the Ultra DMA interface, which all other manufacturers now use as well.

**1997**
IBM announces the world's highest-capacity desktop PC hard disk drive with new breakthrough technology called Giant Magnetoresistive (GMR) heads. Pioneered by scientists at IBM Research, GMR heads will be used in IBM's Deskstar 16GP, a 16.8-gigabyte drive. This brings down the cost of storage to 25 cents per megabyte.

**1998**
IBM announces a 25GB hard drive. The first hard disk drive in 1956 had a capacity of 5MB. IBM's Deskstar 25GP 25GB drive has 5,000 times the capacity of the first drive. It holds either the double-spaced typed text on a stack of paper more than 4,000 feet high, more than six full-length feature films, or 20,000 digital images.

**1998**
The DVD-RAM drive debuts. 5.2GB rewritable capacity on a double-sided cartridge, enough to hold a full-length 2-hour Hollywood movie.

**1999**
The new and improved Ultra DMA-66 interface, which will speed up the EIDE, is announced.

**interleave factor** The number of sectors that pass beneath the heads before the next sector to be read reaches the heads. An optimized interleave factor provides faster reads of data.

**laser** Acronym for Light Amplification by Stimulated Emission of Radiation, a device that produces a narrow, coherent beam of light. *Coherent* means that all the light waves are moving in unison so the beam does not disperse, or spread and become fainter, as ordinary light does.

**low-level formatting** The process of creating sectors on the media so that a drive can store information.

**optical storage** Drives that use lasers to store and read data.

**platter** A glass or aluminum disk covered with a magnetic material on which data is recorded.

**read** Retrieve data from storage, such as a file from a drive.

**read-only** A storage device—drive or memory—to which new data cannot be written.

**read/write head** The drive component that writes data to a drive and reads it, using magnetism or a laser.

**Redundant array of independent disks (RAID)** An arrangement of several hard drives in the same computer that act as if they were a single drive. A RAID is designed to protect against drive failure or improve performance.

**rewritable** A storage device that can erase old data and save new data.

**S.M.A.R.T** Self-monitoring analysis and reporting technology, an automatic process in which a drive monitors itself for early signs of unreliability or failure and includes a warning to the computer's user.

**sector** The smallest area on the surface of platter that can be used to store data.

**seek time** The average time a drive takes for the read/write heads to move to a particular track on the disk.

**sequential access** Reading or writing data in a sequential order as opposed to randomly.

**serial ATA** The latest interconnection for the ATA command protocol.

**settle time** The time it takes for a read/write head to stop vibrating after it arrives at a specific track so that it can read or write data reliably.

**soft error** An error in reading data that can be overcome by repeatedly trying to re-read the data. See **hard error**.

**UDF (universal disk format)** A file system that supports optical discs such as CD, CD-RW, DVD-ROM, and DVD-RAM.

**track** A concentric band on a drive disk created by formatting.

**VFAT** Virtual File Allocation Table in Windows 95 and 98 is a special file where a drive stores a record of the locations of all the files on the disk. Called simply FAT in DOS.

**voice coil motor** An electro-magnetic motor (actuator) used to position the read-write heads. A wire coil is placed in a stationary magnetic field. Passing current through the coil causes a magnetic flux that makes the coil move.

**1999**
IBM introduces the Microdrive, the world's smallest and lightest hard disk drive, revolutionizing the portable device industry, including PDAs, digital cameras, laptops, and MP3 players. The MicroDrive used most often in portable devices fits 340MB in a 1.7×1.4-inch case—enough for more than 55 high-quality, 2-megapixel digital photo images. It weighs less than a double-A battery.

**2000**
Microsoft Windows 2000 requires 100MB for a full installation (including the browser, of course).

**2001**
IBM introduces "pixie dust," IBM's newest industry-leading storage breakthrough. The new material—antiferromagnetically-coupled (AFC) media—quadruples the areal density of current hard disk drive products to surpass 100 billion bits/square inch, a density previously thought impossible.

**2001**
Quinta's Optically Assisted Winchester (OAW) drive appears. The initial version is expected to store about 20 gigabits of data per square inch, or more than eight copies of the 32-volume Encyclopedia Britannica in an area slightly larger than a postage stamp. Price: $1,000 (1 cent per megabyte).

**2003**
The serial ATA hard drive appears, gaining success as the fastest hard drive for consumer PCs.

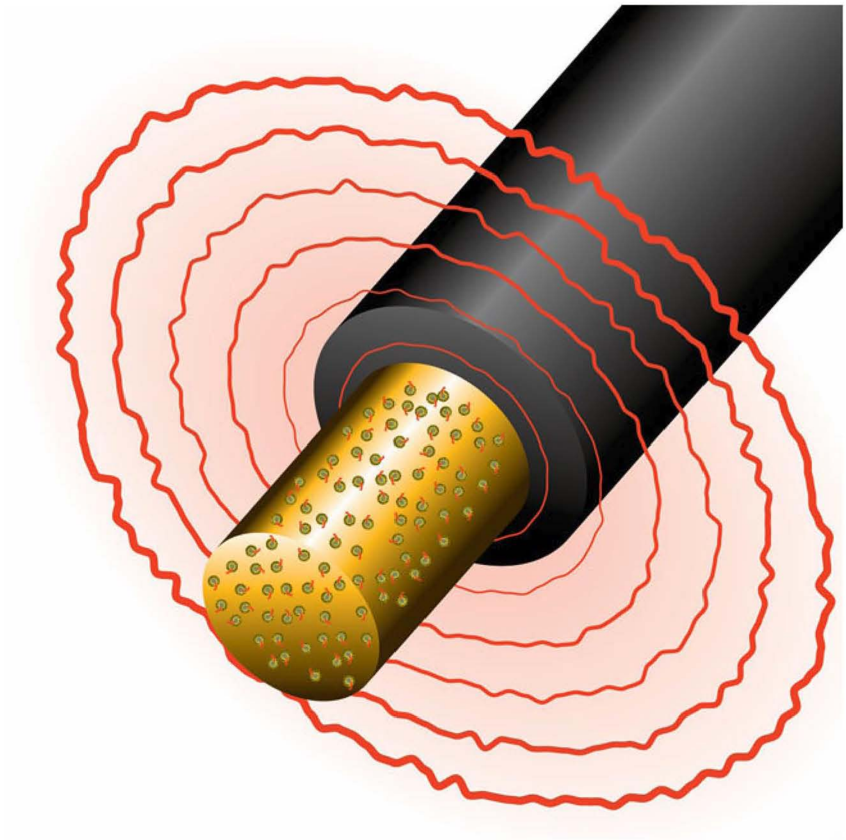
**2005**
A typical desktop hard disk holds 280GB on five platters, enough for 1,500 1GB holographic vacation photos.

**2006**
Sony (Blu-Ray) and Toshiba (HD-DVD) each release their own high-capacity optical disc standard designed specifically for distribution of high-definition video.

**2007**
Hitachi releases the Desktop 7K1000, the first terabyte hard drive on the market.

CHAPTER 10

# How a Computer's Long-Term Memory Works



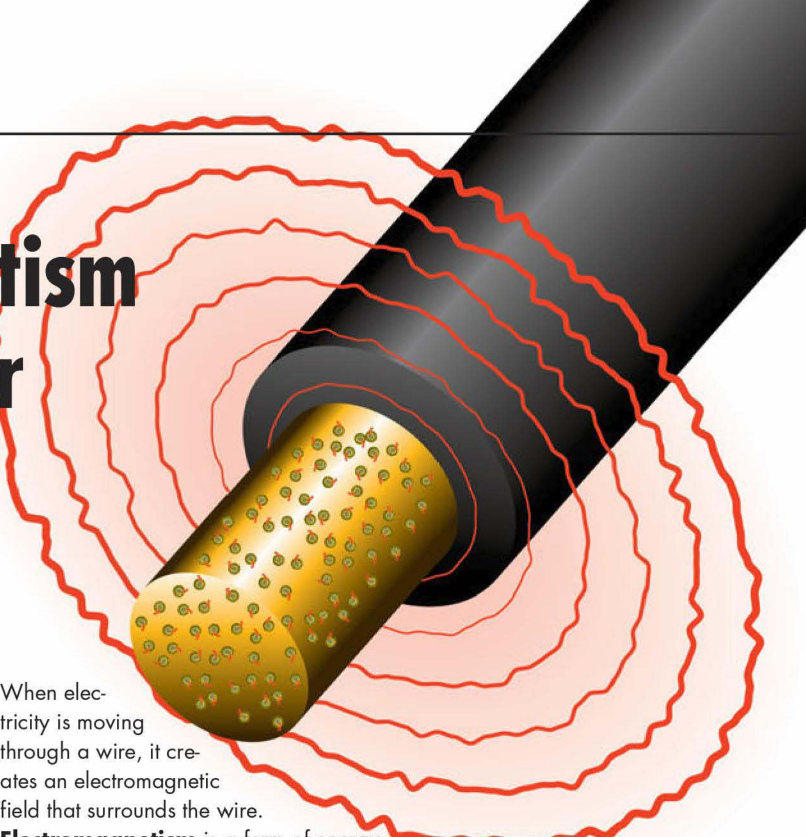
**MAGNETIC** disks are the most common form of permanent data storage. Their capacities can range from a few hundred kilobytes to scores of gigabytes, but they all have some elements in common. For one, the way that a drive's mechanism creates the ones and zeros that make up the binary language of computers might differ, but the goal is the same: to alter microscopically small areas of the disk surface so that some of the areas represent zeros and others represent ones. The new disk uses only those two numbers whether it records a great novel or this week's grocery list.

Another common element among magnetic drives is a scheme that determines how the data on the disk is organized. The computer's operating system, beginning with DOS and continuing with Windows and every other OS, determines the scheme. Many people forget just how long PC hard drives felt DOS's influence, even after Windows 95 took the world by storm. In Windows 95, Windows 98 and Windows Me, the older DOS is still there, it was just hidden beneath Windows's graphic interface. The operating system controls so many of a PC's operations that many PC users forget that DOS stands for **disk operating system** and that, originally, its primary function was to control disk drives.

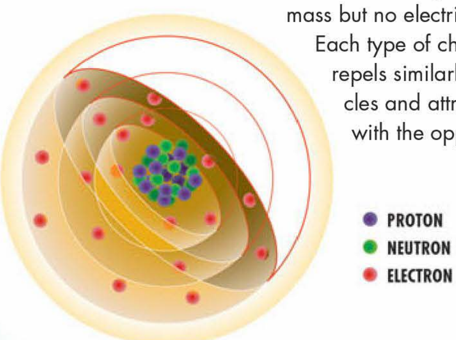
Before any information can be stored on a magnetic disk, the disk must first be **formatted**. Formatting creates a road map that allows the drive to store and find data in an orderly manner. The road map consists of magnetic markers embedded in the magnetic film on the surface of the disk. The codes divide the surfaces of the disk into sectors (pie slices) and tracks (concentric circles). These divisions organize the disk so that data can be recorded in a logical manner and accessed quickly by the read/write heads that move back and forth over the disk as it spins. The number of sectors and tracks that fit on a disk determines the disk's capacity to hold information.

After a disk is formatted, writing or reading even the simplest file is a complicated process. This process involves your software, operating system, the PC's **BIOS (basic input/output system)**, software drivers that tell the operating system how to use add-on hardware such as an external USB flash memory drive, and the mechanism of the disk drive itself.

# How Electromagnetism Reacts with Matter

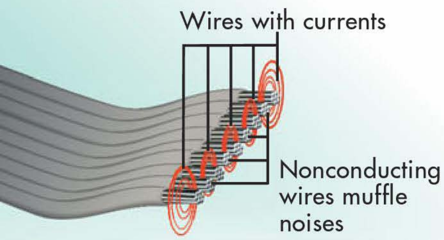


**1** All atoms are made up of positively charged particles called **protons** at the center, or nucleus, of the atom, and the cloud of negatively charged **electrons** that surround the nucleus. (The nucleus also contains **neutrons**, particles that have mass but no electrical charge.) Each type of charged particle repels similarly charged particles and attracts particles with the opposite charge.




**3** When electricity is moving through a wire, it creates an electromagnetic field that surrounds the wire. **Electromagnetism** is a form of energy that spreads in the form of waves. Light, radio signals, microwaves, television signals, and heat are examples of electromagnetism.

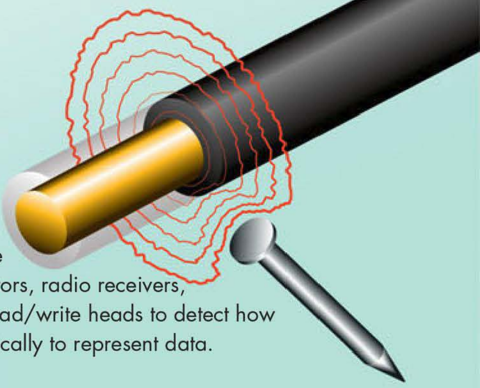
**2** In some atoms, such as copper and aluminum, the attraction is weak between the protons and the electrons in the atoms' outermost layer. In such **conductive** materials, electrons jump freely from one atom to another under the right circumstances. This movement of electrons is **electricity**. In other materials, such as rubber and glass, electrons are more closely bound to their nuclei and do not easily move from one atom to another. These **nonconductive** materials are **insulators**. Still other materials, such as silicon, can act as either conductors or nonconductors under different conditions. These are **semiconductors**, an important component of microchips and transistors.

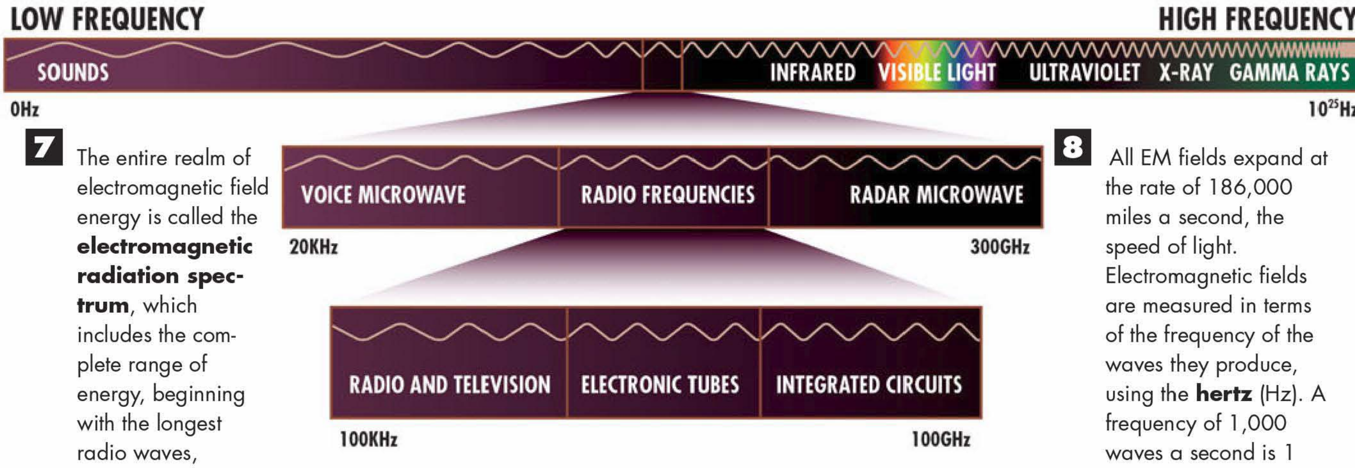



**4** All electrical devices invariably create stray, unwanted electromagnetic fields called **noise**. Static on the radio and snow on a television screen are tangible examples of electrical noise. Much of the electromechanical design of computer components, such as adding nonfunctioning wires to cables, is aimed at reducing the interference of noise. The noisier the environment, the stronger a signal must be to make itself heard over the noise. And the stronger a signal, the more noise it creates that affect other components.

**5** Conversely, when a wire moves through a magnetic field, the interaction creates an electrical current in the wire. **AC** electrical current is produced by generators that are essentially giant coils of wire spinning through a field created by massive magnets.

**6** Similarly, electromagnetic fields are influenced by the presence, proximity, shape, composition, and mass of objects moving through the field. Those changes to the frequency or amplitude of an EM field can be detected and are the basis for metal detectors, radio receivers, and the capability of a computer drive's read/write heads to detect how particles on the disk are arranged magnetically to represent data.

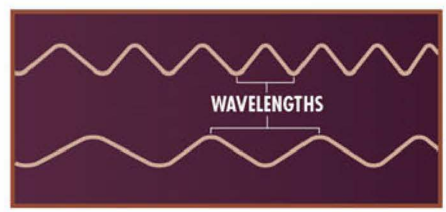


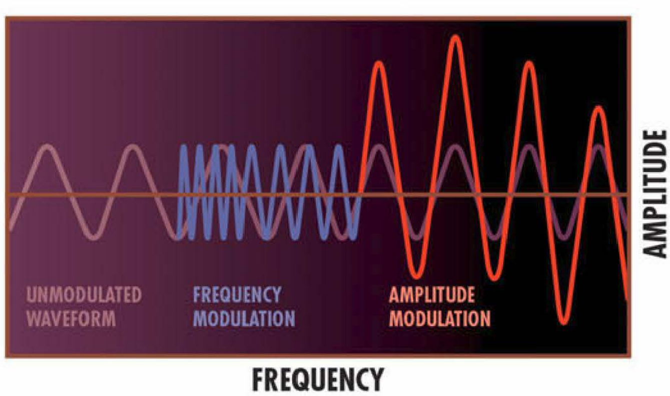



**7** The entire realm of electromagnetic field energy is called the **electromagnetic radiation spectrum**, which includes the complete range of energy, beginning with the longest radio waves, through visible light—a very small part of the spectrum—to the extremely short gamma rays produced by radioactive atoms and major astronomical events, such as novas.

**8** All EM fields expand at the rate of 186,000 miles a second, the speed of light. Electromagnetic fields are measured in terms of the frequency of the waves they produce, using the **hertz** (Hz). A frequency of 1,000 waves a second is 1 kilohertz (kHz). In the case of infrared, visible light, ultraviolet, and gamma radiation, the wavelength is more often specified in **nanometers** (units of $10^{-9}$ meter) or **Angstrom units** (units of $10^{-10}$ meter).

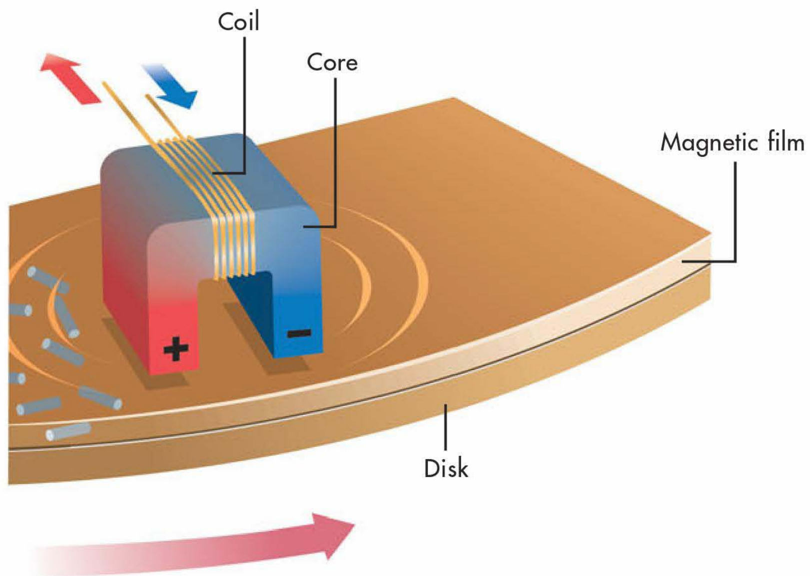
**9** The frequency of an EM field is inversely related to its **wavelength**, the distance between identical points in adjacent waves. The higher the frequency of the signal, the shorter the wavelength. A signal at 100MHz—in the middle of the FM radio broadcast band—has a wavelength of about 10 feet. A signal at 30 gigahertz (GHz)—in the range of radar and microwaves—has a wavelength of a little less than half an inch.



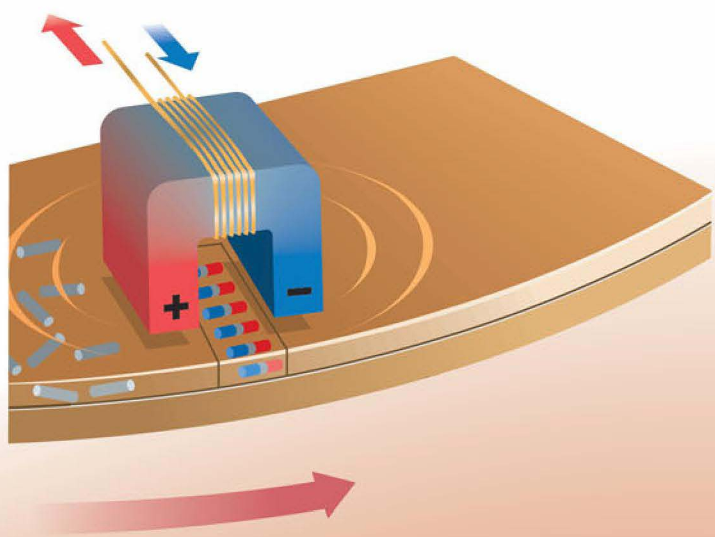




**10** Electromagnetic fields are used as signals to carry data by creating variations in the **waveform**.
**Frequency modulation** (FM) carries data by varying the frequency of a fundamental waveform.
**Amplitude modulation** (AM) varies the strength, or amplitude, of a basic waveform. The amount of data that a signal can carry increases with the frequency of the electromagnetic field creating them. Because there are more variations in one second of higher frequency waves, there are more opportunities to modulate the wave so that it carries data.

# How a Drive Writes and Reads Bits on a Disk

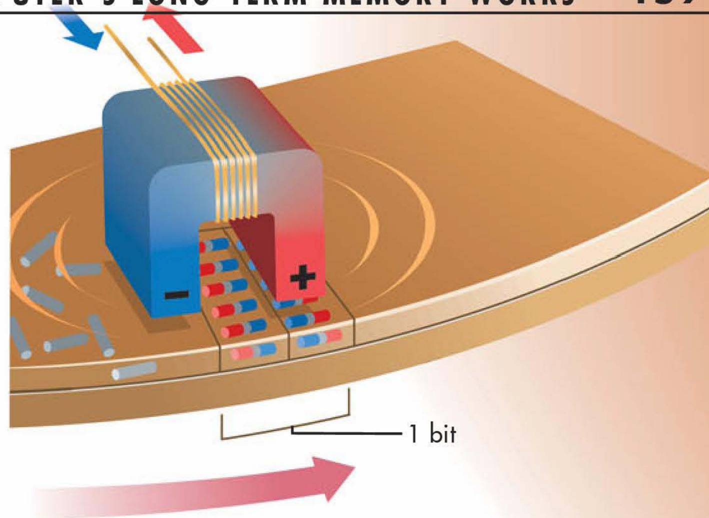



**1** Before a PC writes any data to a magnetic drive, iron particles are scattered in a random pattern within a magnetic film that coats the surface of the disk. The film is similar to the surface of audio and video tapes. To organize the particles into data, electricity pulses through a coil of wire wrapped around an iron core in the drive mechanism's read/write head, which is suspended over the disk's surface. The electricity turns the core into an electromagnet that can magnetize the particles in the coating, much as a child uses a magnet to play with iron filings.

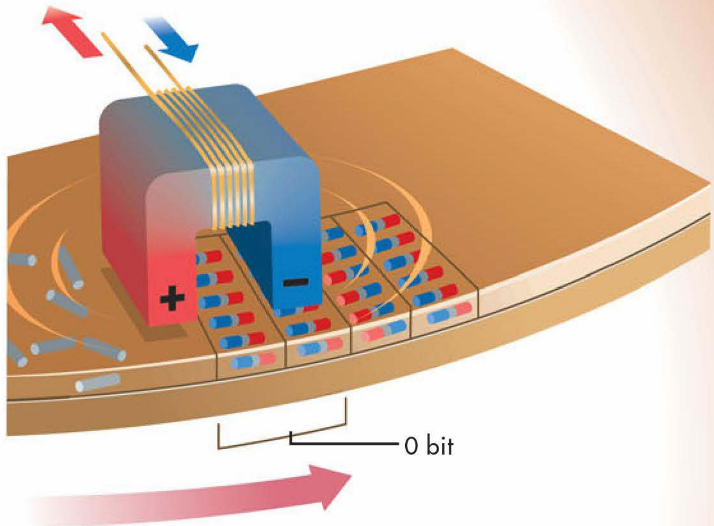


**2** The coil induces a magnetic field in the core as it passes over the disk. The field, in turn, magnetizes the iron particles in the disk coating so their positive poles (red) point toward the negative pole of the read/write head, and their negative poles (blue) point to the head's positive pole.

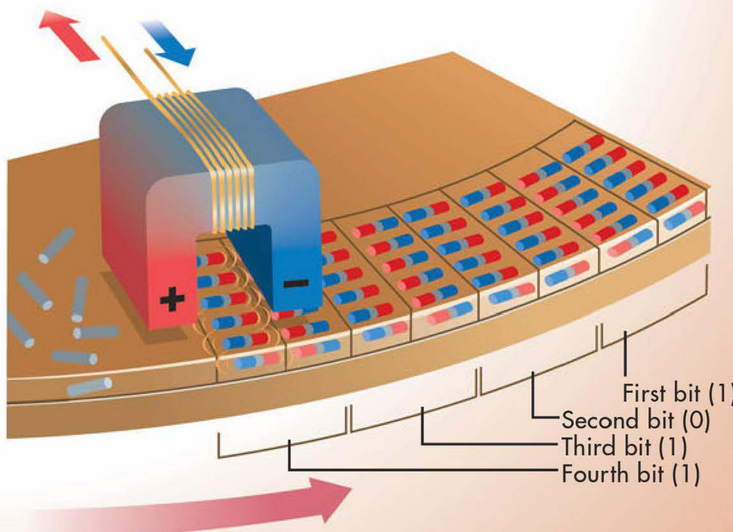
**3** After the head creates one band of aligned, magnetized particles on the revolving disk, a second band is created next to it. Together, the two bands represent the smallest discrete unit of data that a computer can handle—a *bit*. If the bit is to represent a binary 1, after creating the first band, the current in the coil reverses so that the magnetic poles of the core are swapped and the particles in the second band are magnetized in the opposite direction. If the bit is a binary 0, the particles in both bands are aligned in the same direction.




**4** When a second bit is stored, the polarity of its first band is always the opposite of the band preceding it to indicate that it's beginning a new bit. Even the slowest drive takes only a fraction of a second to create each band. The stored bits in the illustration represent the binary numeral 1011, which is 11 in decimal numbers.




**5** To read the data, no current is sent to the read/write head as it passes over the disk. Instead, the magnetic opposite of the writing process happens. The banks of polarized particles in the disk's coating are themselves tiny magnets that create a magnetic field through which the read/write head passes. The movement of the head through the magnetic field generates an electrical current that travels in one direction or the other through the wires leading from the head. The direction the current flows depends on the polarities of the bands. By sensing the changes in direction of the current, the computer can tell whether the read/write head is passing over a 1 or a 0.



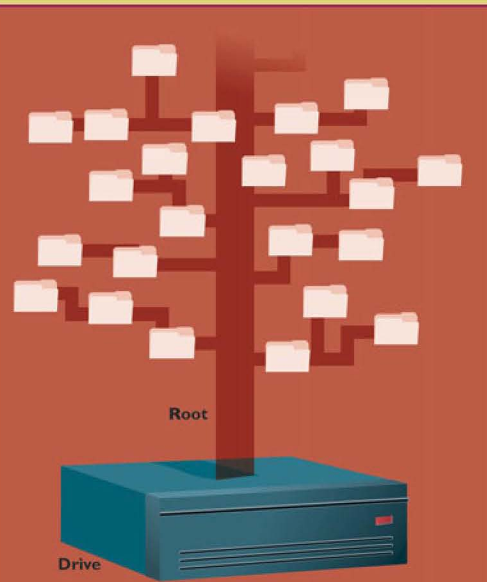
# How a Drive Maps a Disk's Surface

**1** The first task a magnetic drive must accomplish is to format any disk that is used with it so that there is a way to organize and find files saved to the disk. It does this by writing a pattern of ones and zeros onto the surface of the disk—like magnetic signposts. The pattern divides the disk radially into sectors and into concentric circles called **tracks**. As the read/write head moves back and forth over the spinning disks, it reads these magnetic signposts to determine where it is in relation to the data on the disk's surface.

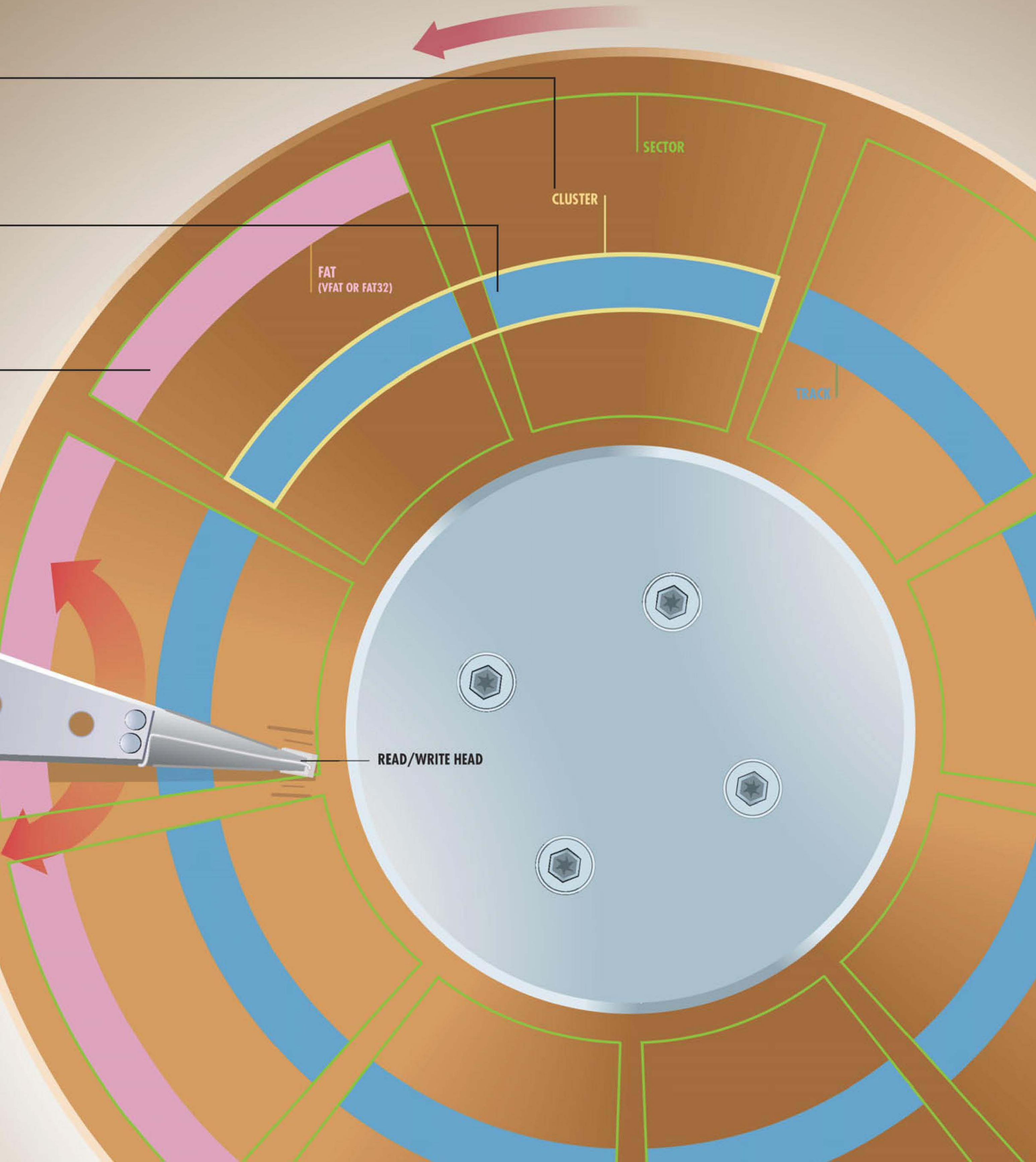
**2** Two or more sectors on a single track make up a **cluster** or **block**. The number of bytes in a cluster varies according to the disk's size and the version of the operating system used to format the disk. A cluster is the minimum unit the operating system uses to store information. Even if a file has a size of only 1 byte, a cluster as large as 32 kilobytes (KB) might be used to contain the file on large drives. The number of sectors and tracks and, therefore, the number of clusters that a drive can create on a disk's surface, determine the capacity of the disk.

**3** The drive creates a special file located in the disk's sector 0. (In the computer world, numbering often begins with 0 instead of 1.) This file is called the **file allocation table**, or **FAT**, in DOS, and the **VFAT (virtual FAT)** in Windows 95/98. VFAT is faster because it allows the computer to read files 32 bits at a time, compared to the 16-bit reads of the older FAT. VFAT also allows the use of filenames up to 255 characters long, compared to the 11 used by DOS. The FATs are where the operating systems store the information about the disk's directory, or folder structure, and which clusters are used to store which files. They also permit clusters of 4KB regardless of disk size. With Windows 98 came FAT32, which allows hard drives larger than 2 gigabytes to be formatted as a single disk. An identical copy of the FAT is kept in another location in case the data in the first version becomes corrupted. Ordinarily, you will never see the contents of the FAT, VFAT, or FAT32. In the NT file system (NTFS), which is used in Windows 2000 and XP, the FAT disappears entirely. Information about a file's clusters is instead stored in each one of those clusters.

### The Computer Filing Cabinet

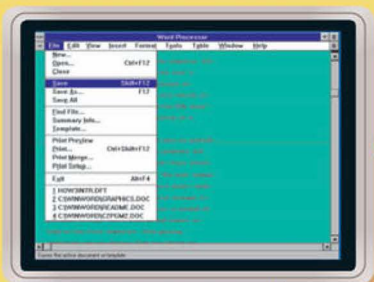



Think of a disk as being a filing cabinet in which you keep all your documents. Each drawer in the cabinet is the equivalent of one of your drives—floppy, hard disk, or optical. On each drive, the first level of organization, called the *root*, contains **directories**, or **folders**—the digital equivalent of a file cabinet's cardboard file folders. Each directory contains the individual files—documents, spreadsheets, graphics, programs—the same way that a drawer's file folders contain individual letters, reports, and other hard copy. One important difference is that it's easy for drive folders to contain other folders, which can contain still more folders, and so on, indefinitely. This directory/folder structure is called a **tree** because a diagram of how it's organized looks like the branching structure of a tree.


SECTOR
CLUSTER
FAT
(VFAT OR FAT32)
TRACK
READ/WRITE HEAD

# How a PC Saves a File to Disk

**1** When you click your mouse to save a file, the program you're using sends a command to Windows, asking the operating system to carry out the steps needed to save the file from RAM, where it's being held temporarily, to disk for permanent storage. For this example, we'll assume you're using a word processor to save a file named Letter to Mom.doc.



**2** Windows modifies the record of the folder (directory) structure stored in the **virtual file allocation table**, or **VFAT** (simply **FAT** in DOS or **FAT32** in Windows 98), to indicate that a file named Letter to Mom.doc will be stored in the current folder, or in another folder if you provide a different directory path.
There is no FAT in Windows NT, 2000, and XP. Location information for each cluster in a file is saved in every cluster that helps make up the file.


VIRTUAL FILE ALLOCATION TABLE

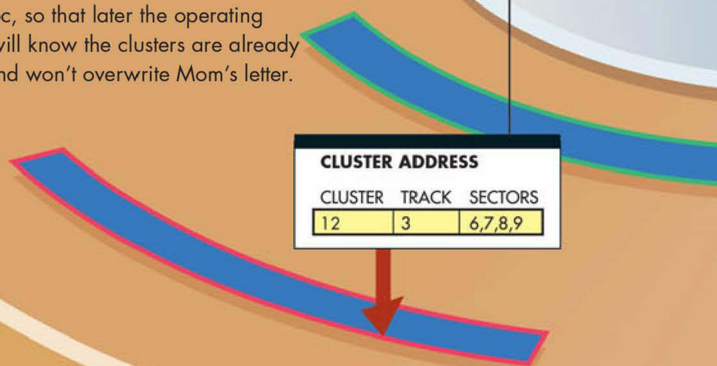
**VIRTUAL FILE ALLOCATION TABLE**

| FILE | CLUSTER |
|---|---|
| Letter to Mom.doc | 3 |
| New Budget.xls | 4 |
| EMPTY | 5 |

**7** Finally, Windows or DOS changes the information contained in the VFAT to mark which clusters contain Letter to Mom.doc, so that later the operating system will know the clusters are already in use and won't overwrite Mom's letter.

**CLUSTER ADDRESS**

| CLUSTER | TRACK | SECTORS |
|---|---|---|
| 12 | 3 | 6,7,8,9 |



## What Happens When You Delete a File?

When you delete a file, the data that makes up the file is not actually changed on the disk. Instead, the operating system changes the information in the VFAT to indicate that the clusters that had been used by that file are now available for reuse by other files. Because the data remains on disk until the clusters are reused, you often can restore—or *undelete*—a file that you've accidentally erased.

**VIRTUAL FILE ALLOCATION TABLE**

Available

| FILE | CLUSTER |
|---|---|
| Letter to Mom.doc | 3 |
| New Budget.xls | 4 |
| EMPTY | 5 |
| Old Budget.xls | 6 |

**3** The operating system also checks the VFAT for the number of a cluster where the VFAT says Windows can save the file without overwriting any other data that's already been saved. In this example, the VFAT tells Windows that Cluster 3 is available to record data.

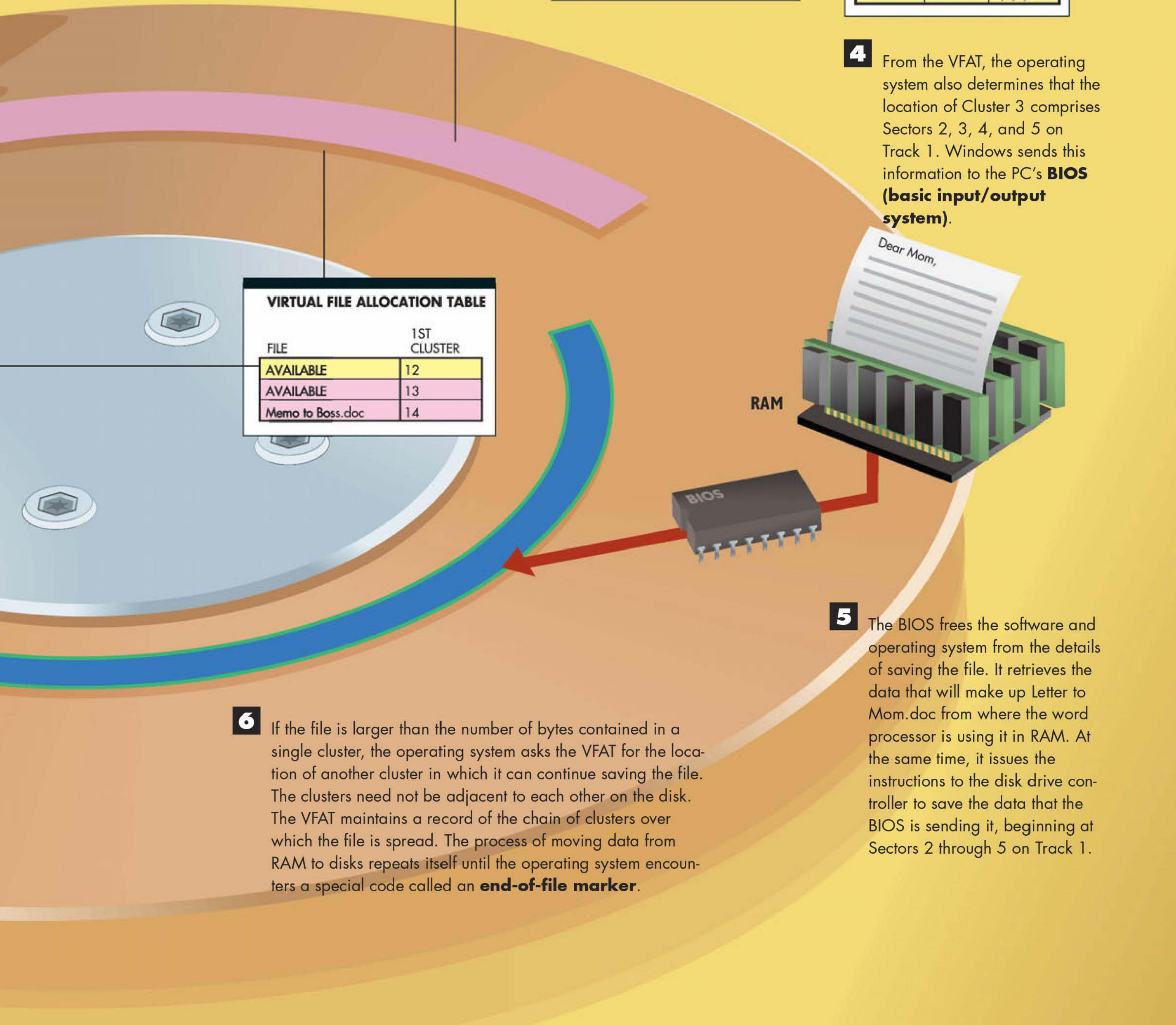
**VIRTUAL FILE ALLOCATION TABLE**

| FILE | 1ST CLUSTER |
|---|---|
| Expenses.xls | 1 |
| Annual Report.doc | 2 |
| AVAILABLE | 3 |
| New Budget.xls | 4 |

**CLUSTER ADDRESS**

| CLUSTER | TRACK | SECTORS |
|---|---|---|
| 3 | 1 | 2,3,4,5 |

**4** From the VFAT, the operating system also determines that the location of Cluster 3 comprises Sectors 2, 3, 4, and 5 on Track 1. Windows sends this information to the PC's **BIOS (basic input/output system)**.




**VIRTUAL FILE ALLOCATION TABLE**

| FILE | 1ST CLUSTER |
|---|---|
| AVAILABLE | 12 |
| AVAILABLE | 13 |
| Memo to Boss.doc | 14 |

**5** The BIOS frees the software and operating system from the details of saving the file. It retrieves the data that will make up Letter to Mom.doc from where the word processor is using it in RAM. At the same time, it issues the instructions to the disk drive controller to save the data that the BIOS is sending it, beginning at Sectors 2 through 5 on Track 1.

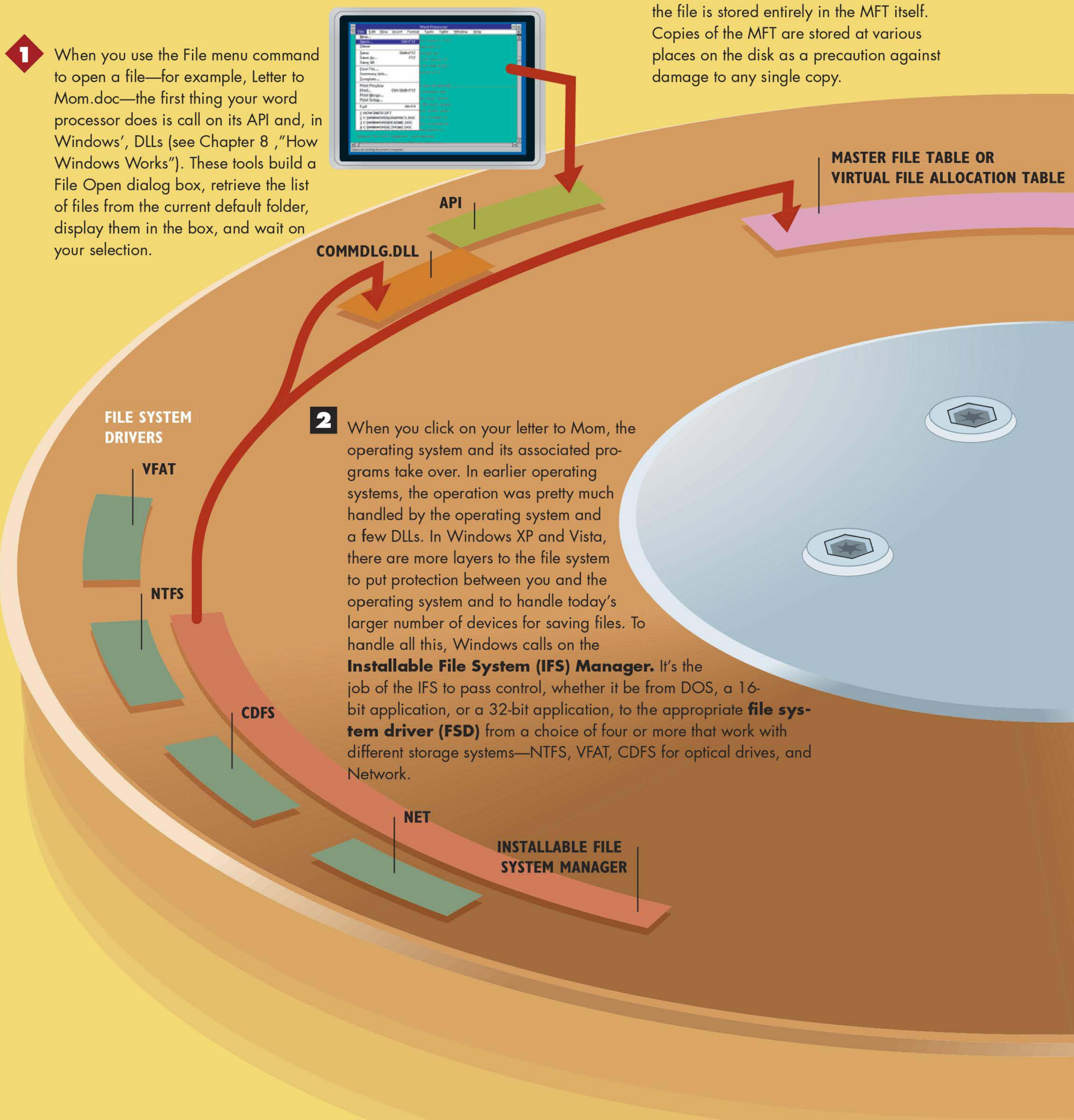
**6** If the file is larger than the number of bytes contained in a single cluster, the operating system asks the VFAT for the location of another cluster in which it can continue saving the file. The clusters need not be adjacent to each other on the disk. The VFAT maintains a record of the chain of clusters over which the file is spread. The process of moving data from RAM to disks repeats itself until the operating system encounters a special code called an **end-of-file marker**.

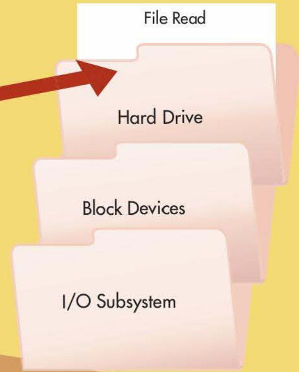
# How a PC Retrieves a File from a Disk

**1** When you use the File menu command to open a file—for example, Letter to Mom.doc—the first thing your word processor does is call on its API and, in Windows', DLLs (see Chapter 8 ,"How Windows Works"). These tools build a File Open dialog box, retrieve the list of files from the current default folder, display them in the box, and wait on your selection.

**2** When you click on your letter to Mom, the operating system and its associated programs take over. In earlier operating systems, the operation was pretty much handled by the operating system and a few DLLs. In Windows XP and Vista, there are more layers to the file system to put protection between you and the operating system and to handle today's larger number of devices for saving files. To handle all this, Windows calls on the **Installable File System (IFS) Manager.** It's the job of the IFS to pass control, whether it be from DOS, a 16-bit application, or a 32-bit application, to the appropriate **file system driver (FSD)** from a choice of four or more that work with different storage systems—NTFS, VFAT, CDFS for optical drives, and Network.

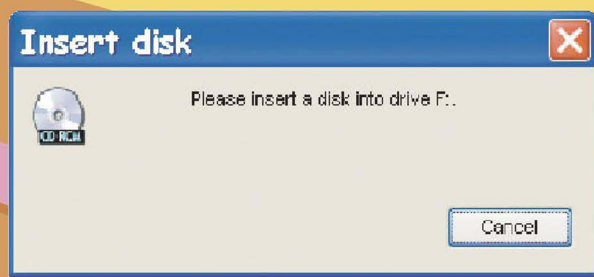
**3** The FSD gets the disk location of the first cluster of the letter for Mom from VFAT or, in Windows XP, from the **MFT (master file table)**, which is the VFAT on steroids, containing more details about a file than its location. In fact, if a file is smaller than 2KB, the file is stored entirely in the MFT itself. Copies of the MFT are stored at various places on the disk as a precaution against damage to any single copy.

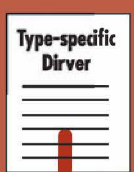





**4** The FSD passes the read command and the info about the file to the next stage, the **IO Subsystem (IOS)**. This is generally an assistant to the FSDs, carrying out such chores as routing messages back and forth between the FSDs and lower, device-specific drivers.

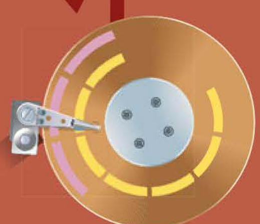
**5** The **volume tracking driver (VTD)** might get into the action if the file you want to read is on a floppy, CD, DVD, or other removable drive. The VTD's only job is to make sure that the correct disc, or disk, is in the correct drive. If you leave a file open on a floppy and remove the floppy disk before you try to save the file, the VTD pops up to tell you to reinsert the right floppy.





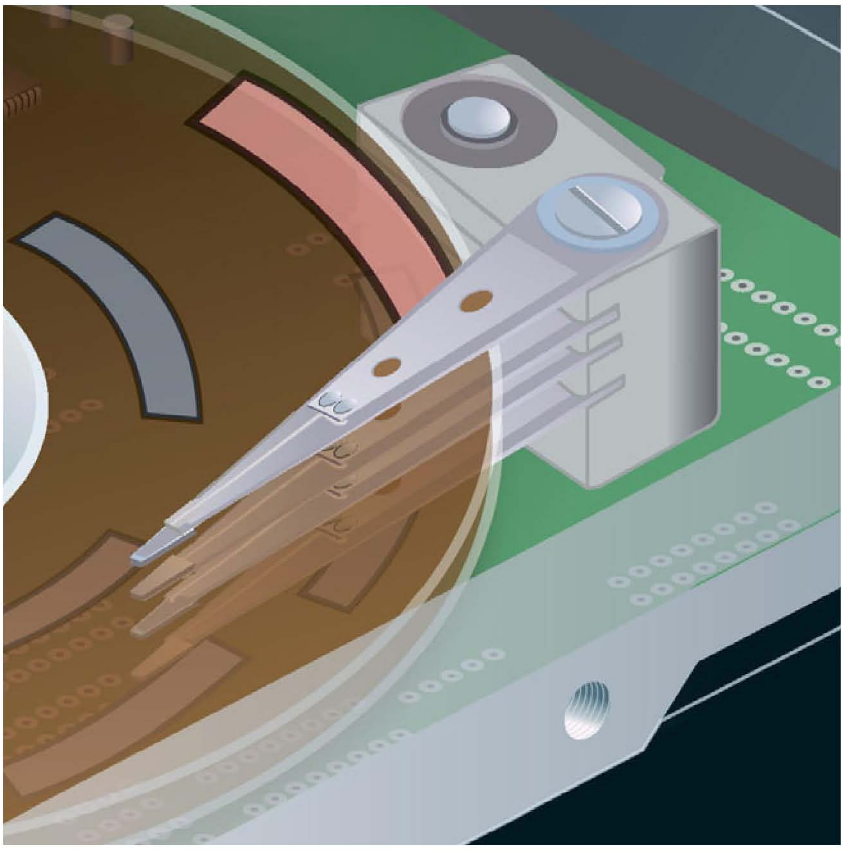
**6** Now the operation passes off to a **type-specific driver (TSD)**. There are individual TSDs for hard drives, floppy drives, network drivers, and any other class of hardware on your PC. A TSD might get help from drivers that the hardware's seller provides and by the **port driver** in Windows to let a message get from the main bus—the motherboard–to the drive's adapter.



**7** Finally, the adapter takes over, moving the read/write heads to the correct series of disk clusters to retrieve the file, which is copied to memory, where you and your computer can work with it.

CHAPTER 11

# How Disk Drives Save Information



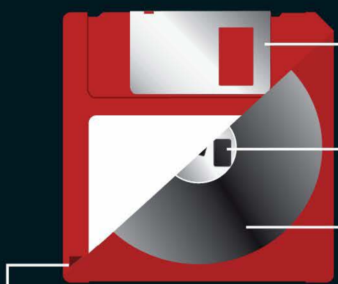
IN more ways than one, a computer's disk drives are where the action is. The platters on which data is stored spin at speeds of up to 15,000 revolutions a minute—167 spins each second. Each time the hard drive is accessed to read or save a file, it causes the read/write heads to burst into a flurry of movement that must be performed with microscopic precision. So exacting are the tolerances in a hard drive—the gaps between the heads and the platters aren't big enough to admit a human hair—it's a wonder the drive can perform its work at all without constant disasters. Instead, it keeps on plugging away as the repository of perhaps years of work—with surprisingly few failures.

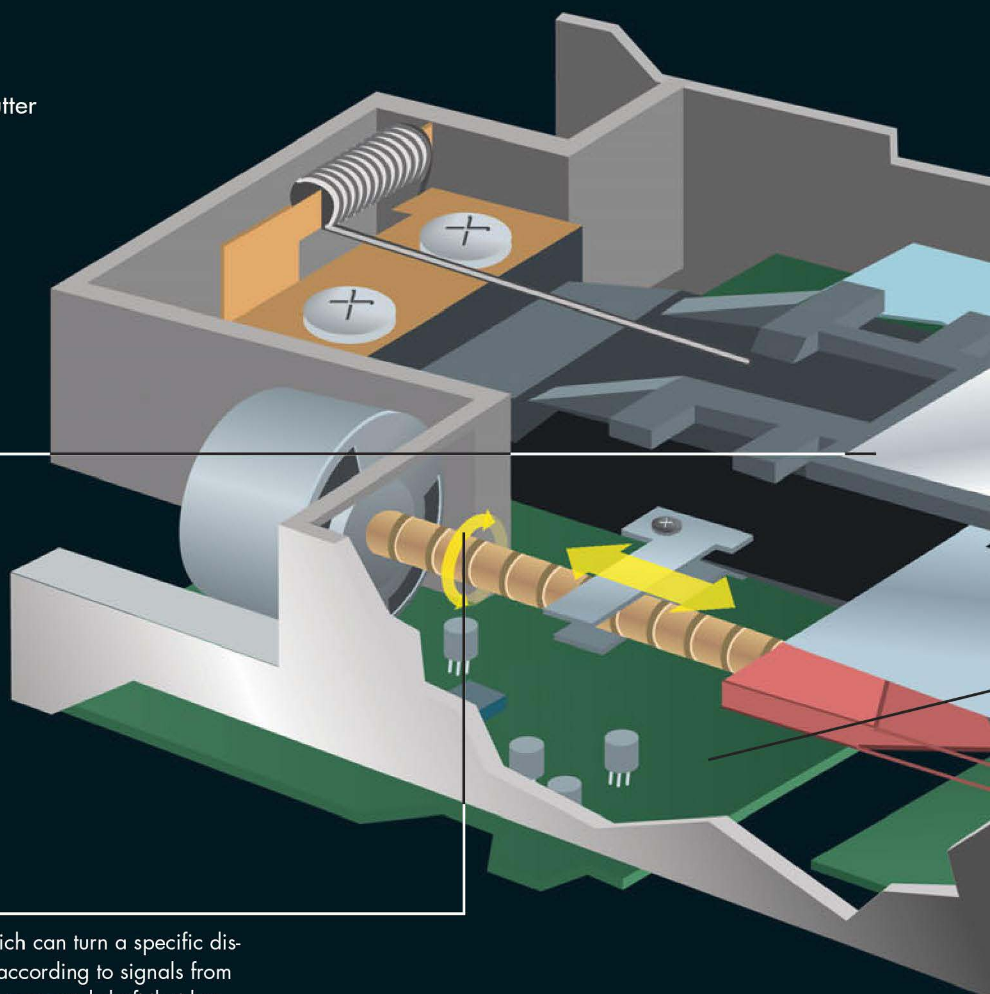
The capacity, size, and performance of hard drives have changed dramatically since the introduction in the early 1980s of the first IBM XT with a hard drive. Back then, a capacity of 10 megabytes was considered generous. The hard drive was 3 to 4 inches thick and filled a 5.25-inch drive bay. An access time of 87 milliseconds was warp speed compared to the access times of floppy drives. A decade later, hard drives that hold more than 1000 gigabytes, smaller than a 3.5-inch floppy drive, and with seek times of 8.5 milliseconds are comparably inexpensive and commonplace. The best part is that it's a certainty the size and prices of drives will continue to decrease while their capacities increase. Who says you can't have a win-win situation?

Amid super fast, super big hard drives, writable CD and DVD drives, and all the other new high-tech marvels, it's hard to get excited about the common floppy drive. After all, it's slow, and a floppy disk doesn't store much information compared to… well, compared to any other type of disk. When the size of software is measured in tens of gigabytes, it's almost unheard of, these days, to see a program distributed on floppy instead of a CD or DVD. In fact, the floppy drive's lifetime is all but over. A little more than twenty years after the 3.5-inch floppy drive was born, today PCs are sold without any type of floppy. Rewritable CD and DVD drives and USB-based flash memory drives, combined with networking that lets you exchange files with others without resorting to any type of drives, have made the floppy obsolete. Oh, you still find them on many new PCs—but not for long.

For all its deficiencies, the floppy drive was an underappreciated wonder of its time. Think of it: An entire book full of information can be contained on a cheap disk that you can slip into your pocket. Until the advent of small, cheap USB-based flash drives, floppy drives were still found on nearly every PC, making them a sure and convenient way to get small amounts of data from one PC to another. No communication lines, networks, gateways, or infrared links were needed; just pull the floppy out of one machine and slip it into another. Today, USB-based flash drives have largely replaced floppy drives for these purposes, but they can't replace the venerable floppy's legacy.

# How a Floppy Drive Stores a Little



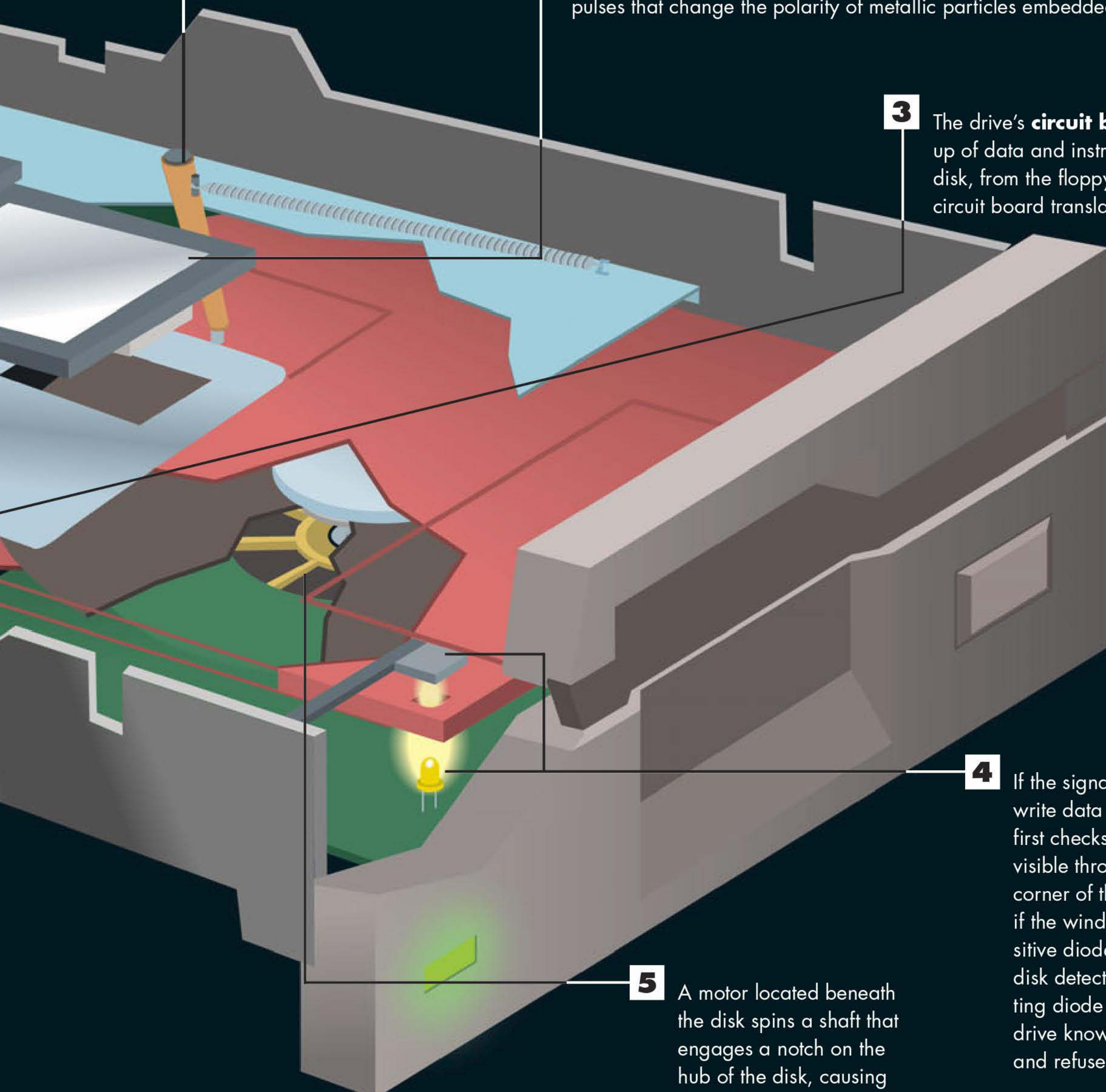
**7** When the heads are in the correct position, electrical impulses create a magnetic field in one of the heads to record data to either the top or bottom surface of the disk. When the heads are reading data, they react to magnetic fields generated by the metallic particles on the disk by sending electrical signals to the computer.

**6** A **stepper motor**—which can turn a specific distance in either direction according to signals from the circuit board—moves a second shaft that has a spiral groove cut into it. An arm attached to the read/write heads rests inside the shaft's groove. As the shaft turns, the arm moves back and forth, positioning the read/write heads over the disk.

**1** When you push a 3.5-inch floppy disk into the drive, the floppy presses against a system of levers. One lever opens the floppy's shutter to expose the **cookie**—a thin Mylar disk coated on either side with a magnetic material similar to the coating on a cassette tape that can record data.

**2** Other levers and gears move two **read/write heads** until they barely touch the cookie on either side. The heads, using tiny electromagnets, generate magnetic pulses that change the polarity of metallic particles embedded in the disk's coating.

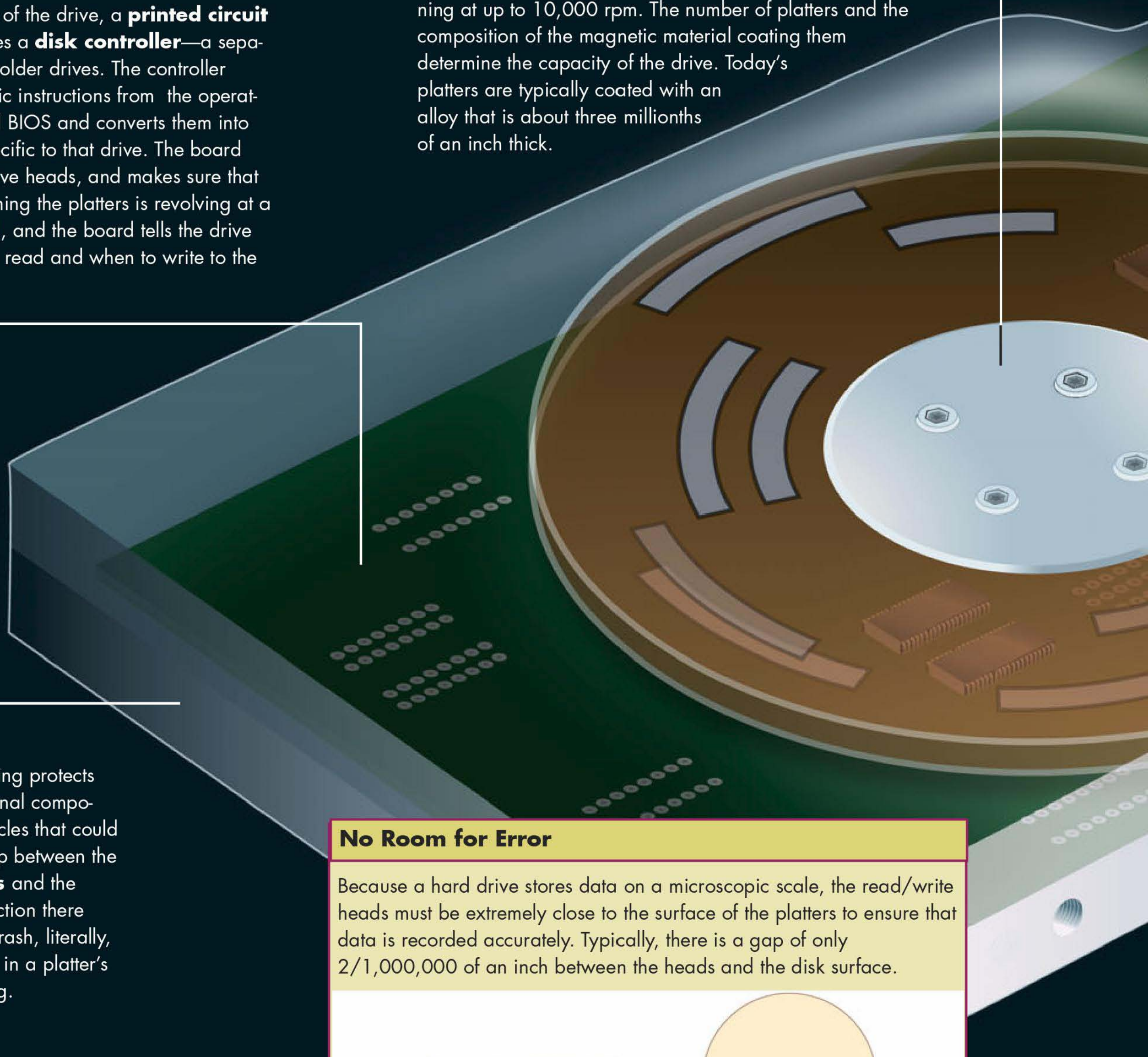
**3** The drive's **circuit board** receives signals, made up of data and instructions for writing that data to disk, from the floppy drive's controller board. The circuit board translates the instructions into electrical signals that control the movement of the disk and the read/write heads.



**4** If the signals include instructions to write data to the disk, the circuit board first checks to be sure that no light is visible through a small window in one corner of the disk's housing. However, if the window is open and a photo-sensitive diode on the opposite side of the disk detects a beam from a light-emitting diode shining through the hole, the drive knows the disk is write-protected and refuses to record new data.

**5** A motor located beneath the disk spins a shaft that engages a notch on the hub of the disk, causing the disk to spin.

# How a Hard Drive Stores a Lot



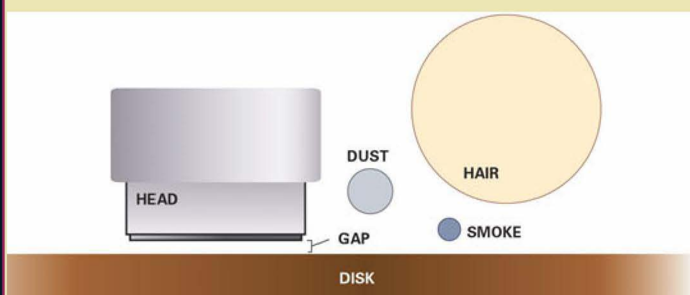
**1** A sealed metal housing protects the hard drive's internal components from dust particles that could block the narrow gap between the **read/write heads** and the **platters**. An obstruction there causes the drive to crash, literally, by plowing a furrow in a platter's thin magnetic coating.

**2** On the bottom of the drive, a **printed circuit board** includes a **disk controller**—a separate board for older drives. The controller receives generic instructions from the operating system and BIOS and converts them into commands specific to that drive. The board controls the drive heads, and makes sure that the spindle turning the platters is revolving at a constant speed, and the board tells the drive heads when to read and when to write to the disk.

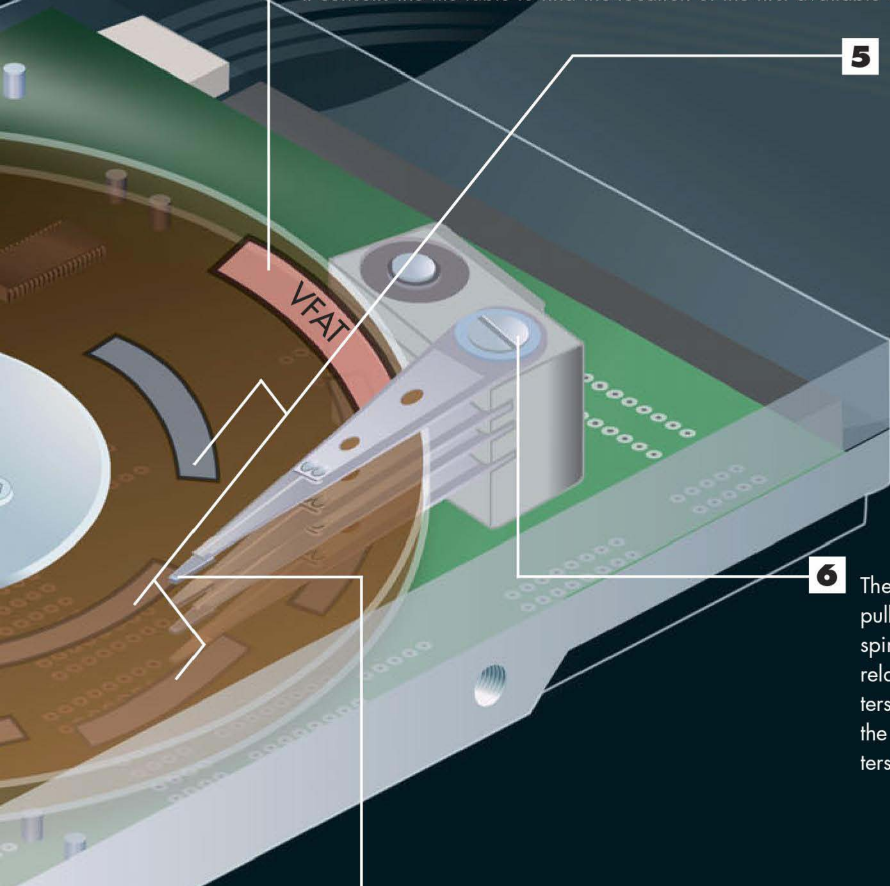
**3** A **spindle** connected to an electrical motor spins as many as eight magnetically coated platters (made of metal or glass), spinning at up to 10,000 rpm. The number of platters and the composition of the magnetic material coating them determine the capacity of the drive. Today's platters are typically coated with an alloy that is about three millionths of an inch thick.

### No Room for Error

Because a hard drive stores data on a microscopic scale, the read/write heads must be extremely close to the surface of the platters to ensure that data is recorded accurately. Typically, there is a gap of only 2/1,000,000 of an inch between the heads and the disk surface.



**4** When your software instructs the operating system to read or write a file, the operating system first checks a file on the drive called the **file allocation table**, also known as **FAT**, **VFAT**, **FAT32**, or **MFT** (**master file table**), depending on the version of Windows you're using. All the tables keep a record of the locations of the clusters that make up all the files on the drive. If the software is writing a new file, it consults the file table to find the location of the first available cluster.

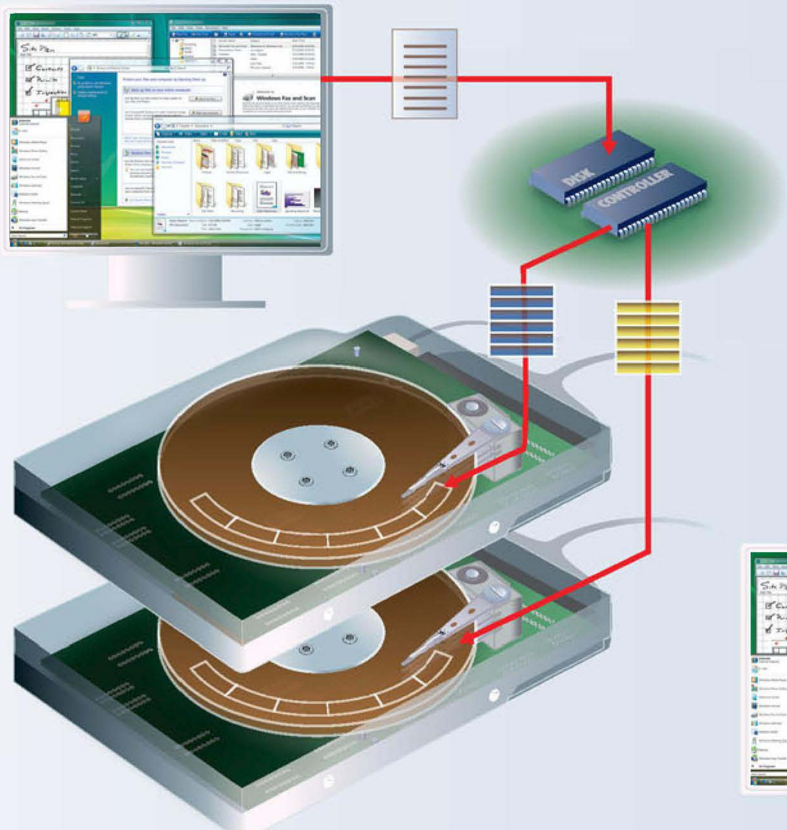



**5** A drive stores a file beginning in the first cluster it finds listed as free in the file table. A single file might be strewn among hundreds of separate clusters scattered across several platters. To read or write a file, the processor passes that command along with the file's name to the drive's built-in **disk controller**.

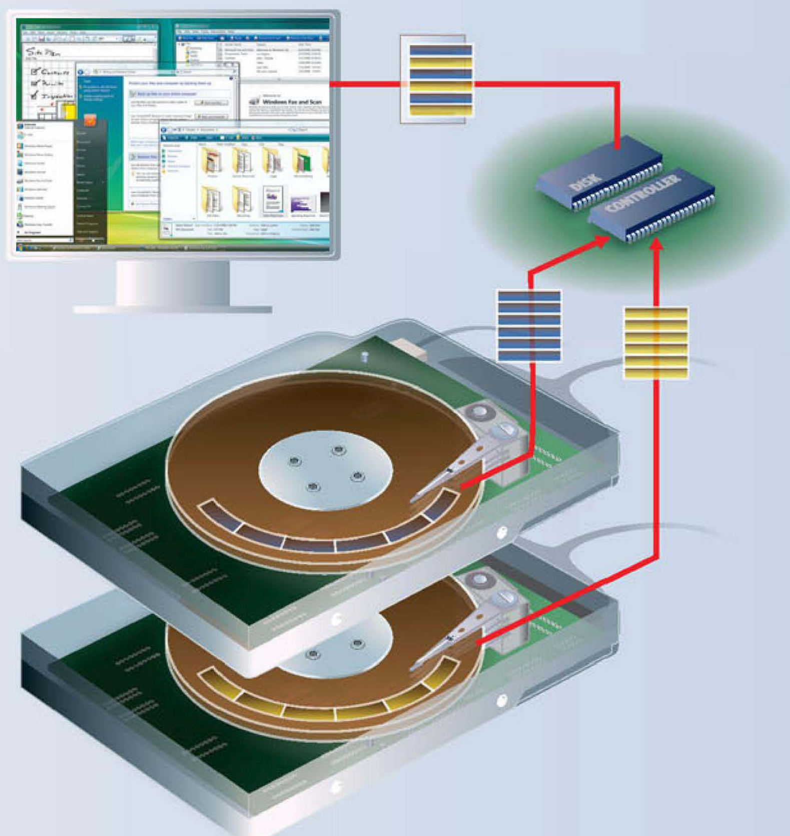
**6** The controller instructs the drive's **head actuator** to push and pull the gang of **read/write arms** across the surfaces of the spinning platters with critical precision as they move among related clusters that can be widely scattered throughout the platters. It aligns the read/write heads on the tips of the arms with the tracks that lie in concentric circles on the surface of the platters.

**7** The heads write the data coming from the disk controller to the platters by aligning the magnetic fields of particles on the platters' surfaces; the heads read data by detecting the polarities of particles that have already been aligned. After the controller writes a new file to the disk, it sends the read/write heads back to the file table, where it records a list of the file's clusters.

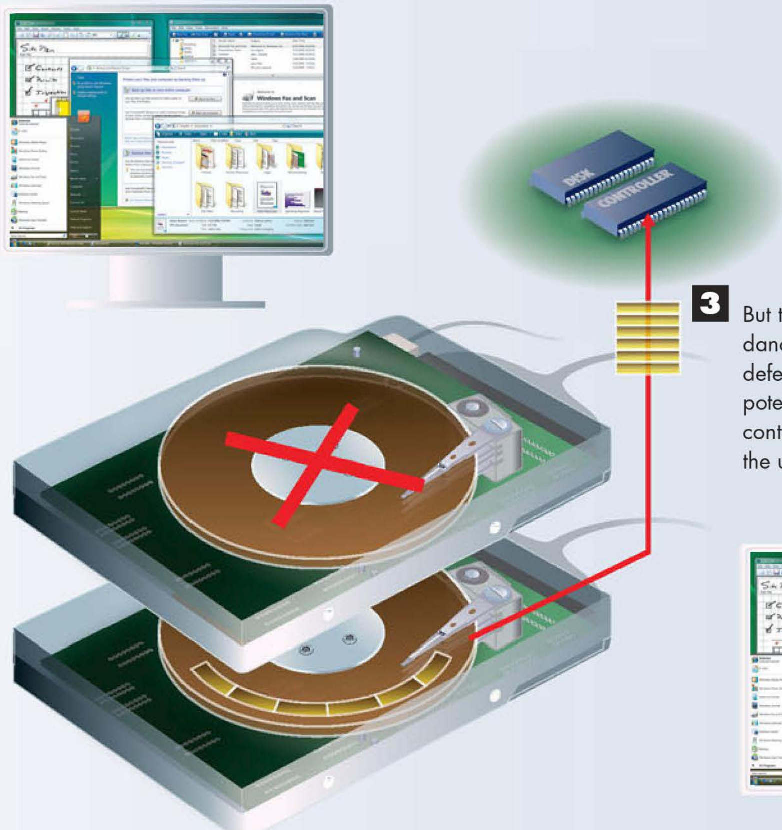
# How a Mirrored Drive Array Protects Files



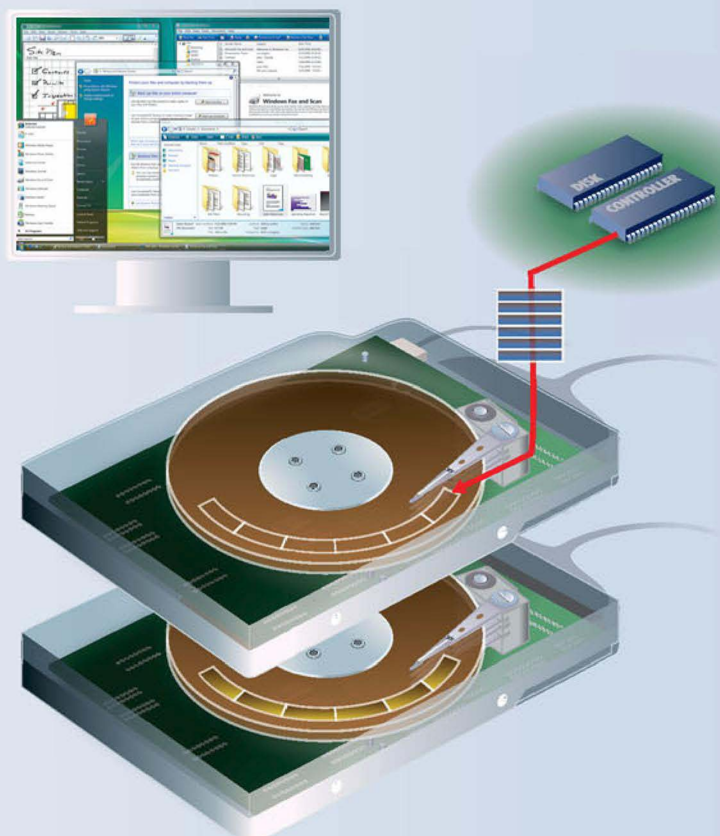

**1** **Drive arrays** work on the simple theory that if one hard drive is a good thing, two hard drives are four times as good. A drive array or **RAID**—for **redundant array of independent**, or **inexpensive**, **drives**—can speed up the performance of a computer's biggest bottleneck, disk drives, or it can provide second-by-second protection against a disk crash. Or both; It depends on how an array is configured. On computer systems where data integrity is more important than speed and there are only two hard drives, the best solution is a **mirrored drive array**, or **RAID 1**. The **RAID controller**—a replacement for the ordinary disk controller with software to handle arrays—writes every file to two or more drives simultaneously. By using multiple hard drives configured so the operating system thinks they are only a single drive, you get greater protection from data loss.




**2** When a file is to be read from a mirrored array, the controller reads alternate file clusters simultaneously from each of the drives and pieces them together for delivery to the PC. This process makes reads faster. How fast depends on the number of drives in the array. If two drives are mirrored, read time is cut approximately in half; three mirrored drives reduce read time to about one-third that of a single drive.

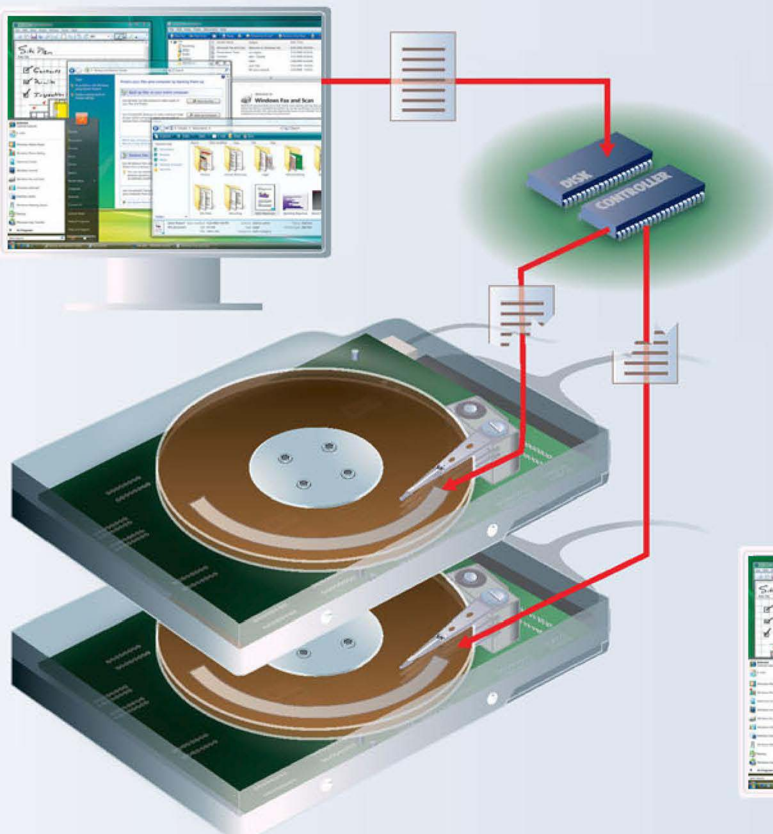



**3** But the real purpose of a RAID 1 is not speed. It's redundancy. In case of a read failure—caused by either a defect on the surface of one of the disks or platters or a potentially more serious crash of one of the drives—the controller simply reads the intact version of the file from the undamaged drive.

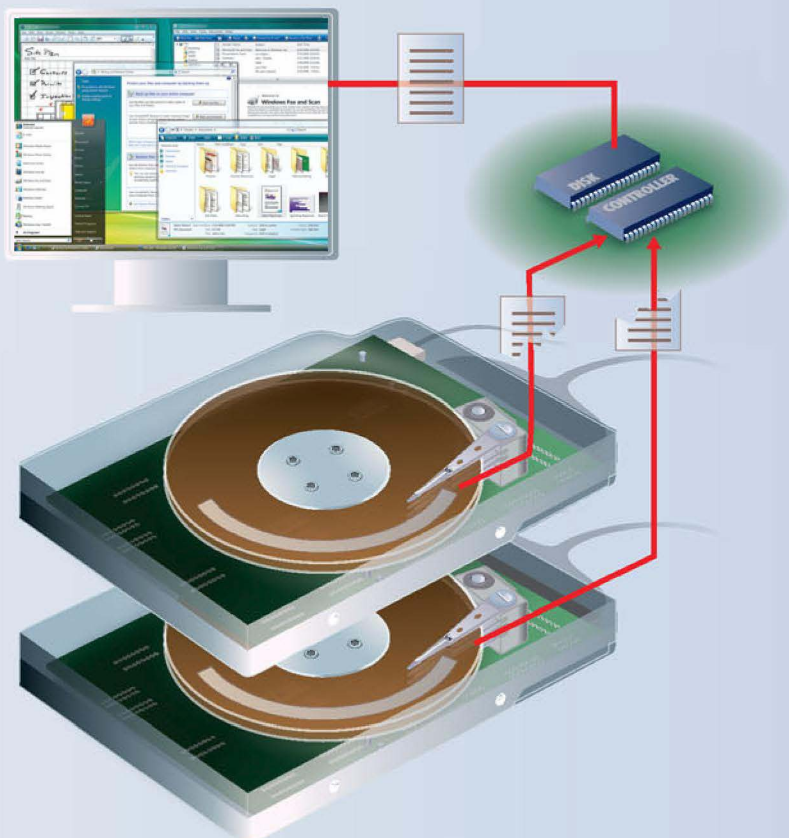



**4** If the read failure is caused by a media defect and the rest of the drive is unscathed, the controller automatically reads the data from the intact copy of the file on the other drive and writes it to a new, undamaged area on the drive on which the defect occurred.

# How a Striped Drive Array Boosts Performance

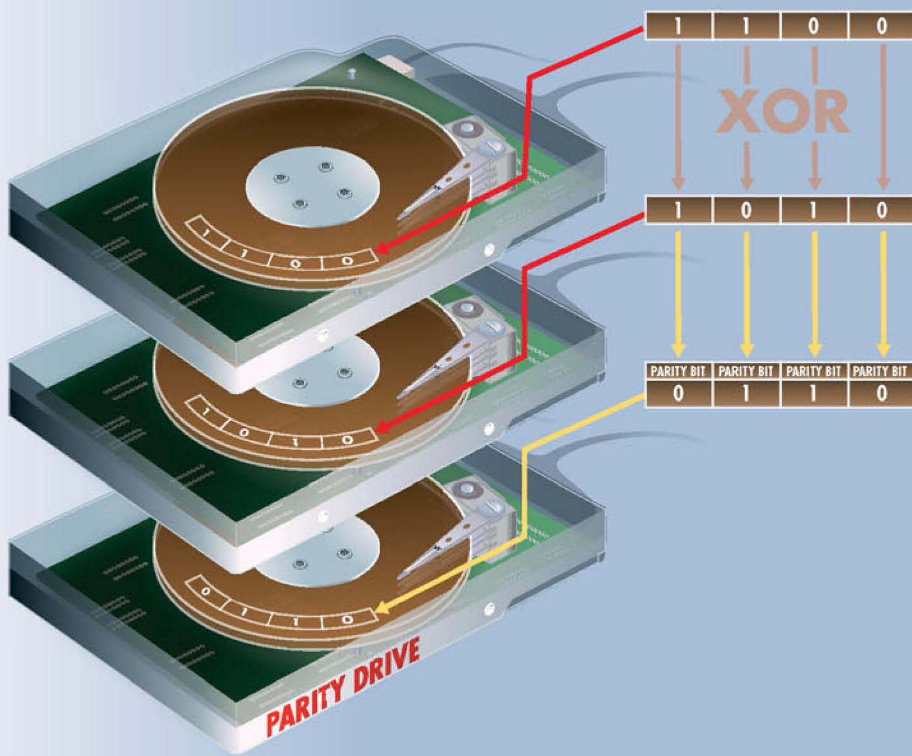


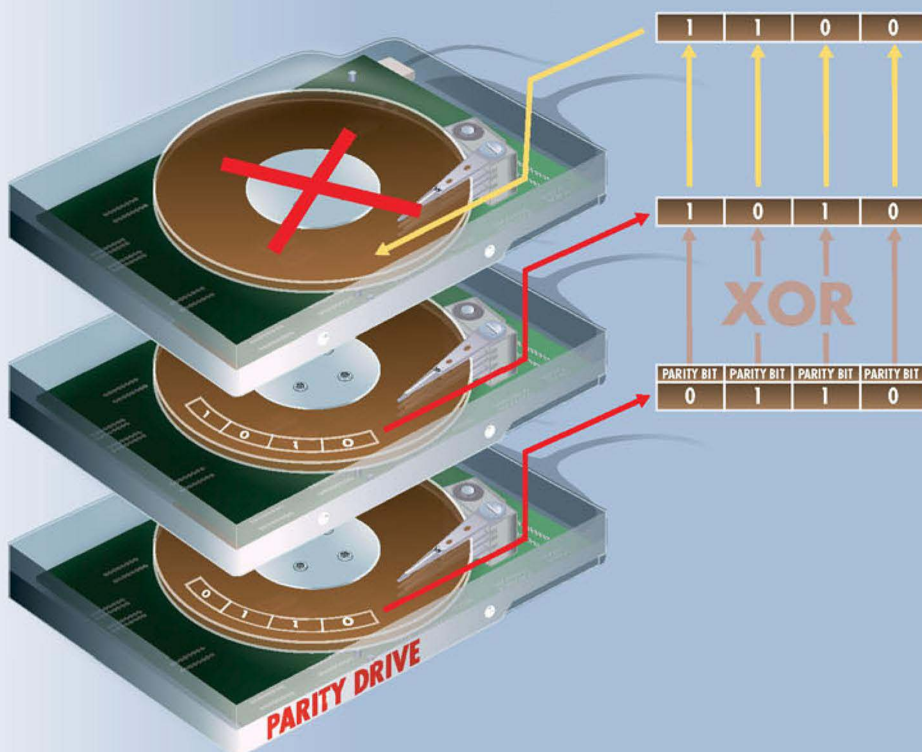

**1** For faster reads and writes, a RAID is set up as a **striped array** of two or more hard disks. When a file is saved, the RAID controller divides the file into several pieces and writes different sections to different drives at the same time.




**2** When a file is read from a striped drive array, the controller simultaneously pulls each part of the file from the different drives on which it is stored, a faster way of retrieving the information. The controller pieces together the parts of the file in the right order when it writes the file to RAM.

**3** If a striped array, also called a **RAID 0**, has more than two drives, it can be configured as a **striped array with parity**, or **RAID 4**. With this array, the controller writes striped data to all but one of the drives and uses the remaining drive for **error checking**. The controller's array software performs a **Boolean XOR** operation on the data written to the drives and writes the result, often called a **parity bit**, to the remaining drive. An XOR operation results in a 0 bit whenever two like bits are compared and a 1 bit whenever two dissimilar bits are compared. For example, XORing each bit in the two binary numbers 1100 and 1010 yields the parity 0110. If more than three drives are in an array, the first two drives are XORed and then that result is XORed with the next drive, and so on, until all the drives containing data have been XORed and the final result is written to the parity drive.

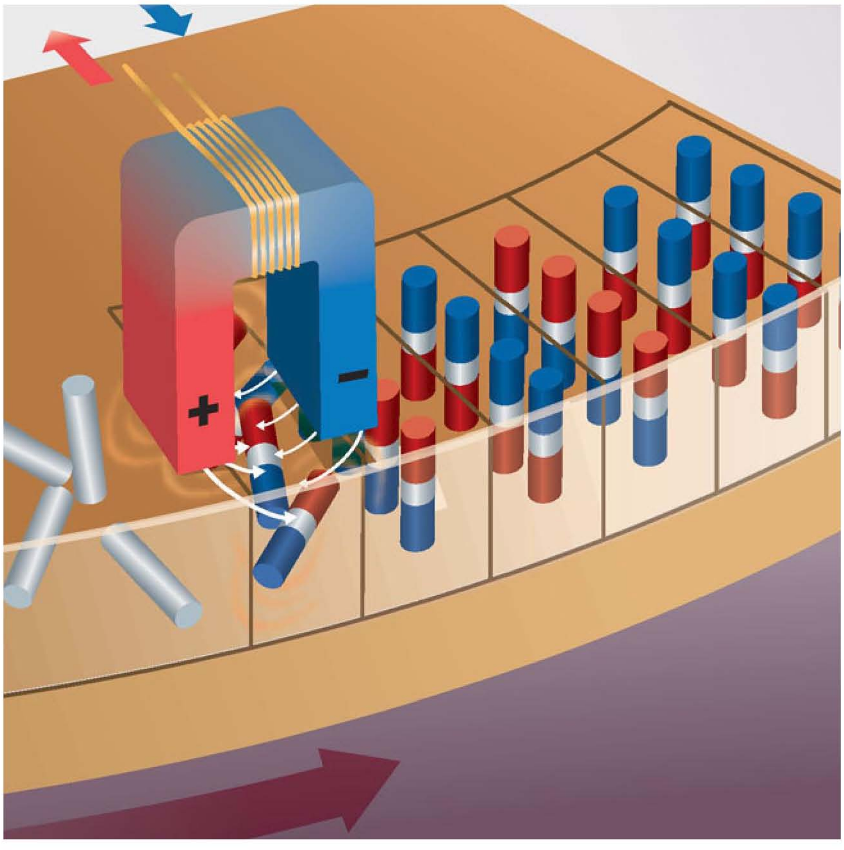



**4** Later, if one of the files is damaged or if one of the drives crashes, the controller performs a reverse XOR operation. By comparing the undamaged bits with the parity bits, the controller can deduce whether the missing bits are 0s or 1s. The information can also be used to repair data caused by minor media defects, allowing the controller to detect and repair flaws on the fly.

CHAPTER

# 12 How Little Things Make Drives Faster and Bigger



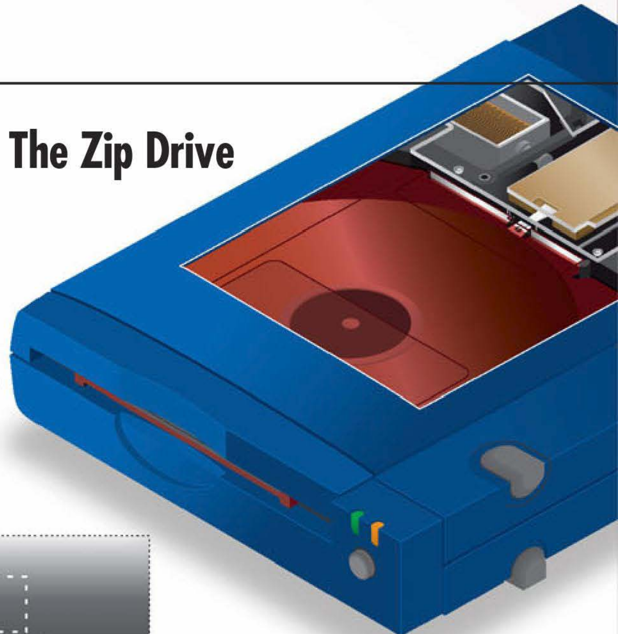
**THE** common image of technological breakthroughs—the wild-haired scientist rising from his test tubes and proclaiming, "Eureka!"—is exciting when it does occur, but that rarely actually happens. Most technological advances are made less by momentous discoveries than by everyday plodding toward incremental gains.

The fundamental technology in personal computers has not changed in the quarter of a century PCs have been changing our lives. Despite the hyping of *Matrix*-like direct links to computers, we're still using keyboards and mice. Fifteen years ago I wrote about how computer data would be stored in holographic cubes of crystal bombarded by laser beams. Yet I was still excited this year when I bought a new hard drive even though it uses electromagnets to write information to metal platters, the same as hard drives have done for decades. But a couple of decades ago, the biggest hard drive you could get for a PC stored 10MB of data in a drive heavier than one of today's laptops. My new drive holds 25,000—*twenty-five thousand*—times as much data, and it's less than half as small. Why isn't that something to shout "Eureka!" about?

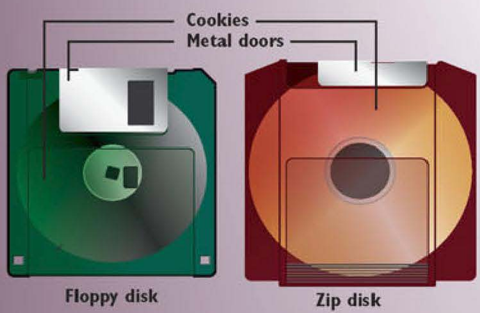
Because all that progress has been made in baby steps. All science progress is 99% incremental—a tweak in design here, a bump in power there, a goose to speed over here, all occasionally given a huge, mind-bending boost by a theory of evolution, the unraveling of DNA, or the invention of the laser. The big developments might pull in the headlines like a black hole, but it's the constant small, day-to-day improvements that have the most immediate impact on what we do. In this chapter we'll look at some less-than-monumental ways in which computer storage—always the laggard in computing speed—has constantly improved by tids and bits.

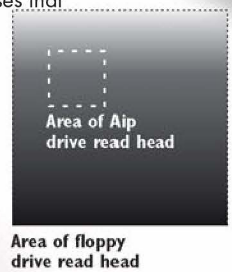
# How Small Changes in Drives Get Big Results

## The Zip Drive

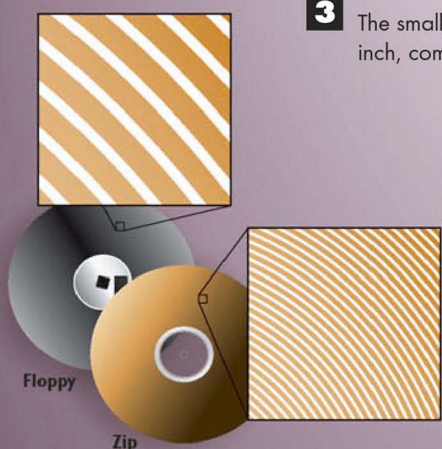


1 Although the Iomega Zip drive lost its popularity to portable hard drives and flash memory, compared to its cousin, the floppy drive, the Zip remains a good example of how small improvements pay off in big benefits in storage capacity. A disk for a Zip drive looks similar to an ordinary floppy disk. It's about the same size, although slightly thicker. And yet a normal 3.5-inch floppy disk can hold a maximum of only 1.4MB of data. The Zip disk holds up to 750MB and accesses that data at speeds that rival low-end hard drives. How does Zip do it?



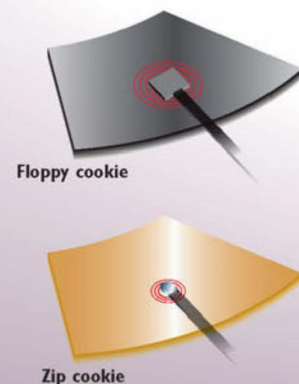



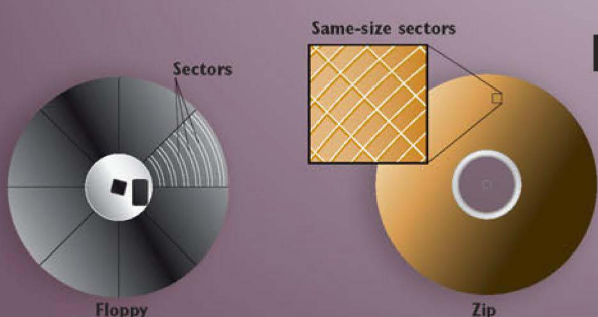

2 The read/write heads are about one-tenth the size of a floppy drive's heads—closer to the size of read/write heads in a hard drive.



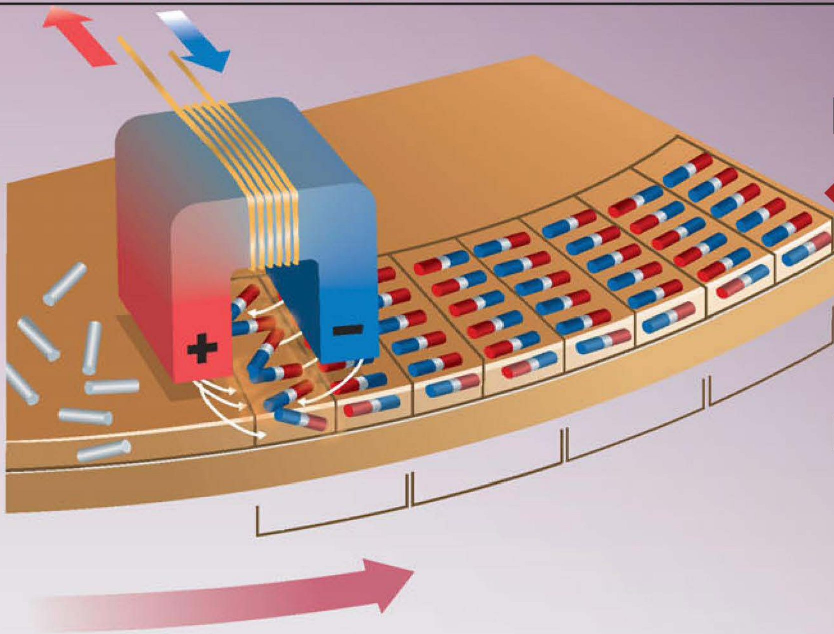

3 The smaller heads allow the Zip drive to write data using 2,118 tracks per inch, compared to the 135 tracks per inch on a floppy disk.

4 Further storage capacity comes from coating the cookie with the same magnetic particles that are used in S-VHS video tape. The particles have a higher **energy level**, which means they are not as easily magnetized. As a result, magnetic fields from the write head affect a far smaller area than a floppy drive does. The smaller the surface area needed to write a 0 or 1 bit, the more bits can be packed onto the same track.




5 A conventional floppy is divided into sectors **radially**. There are as many sectors in the outmost track as there are in the innermost. As a result, the outer tracks use up more surface area than the inner tracks–a waste of the recording surface. A Zip drive, along with hard drives, uses **zone recording**, so that the same recording density is used throughout the disk. This results in more sectors per track as the heads move toward the outer edge of the cookie.



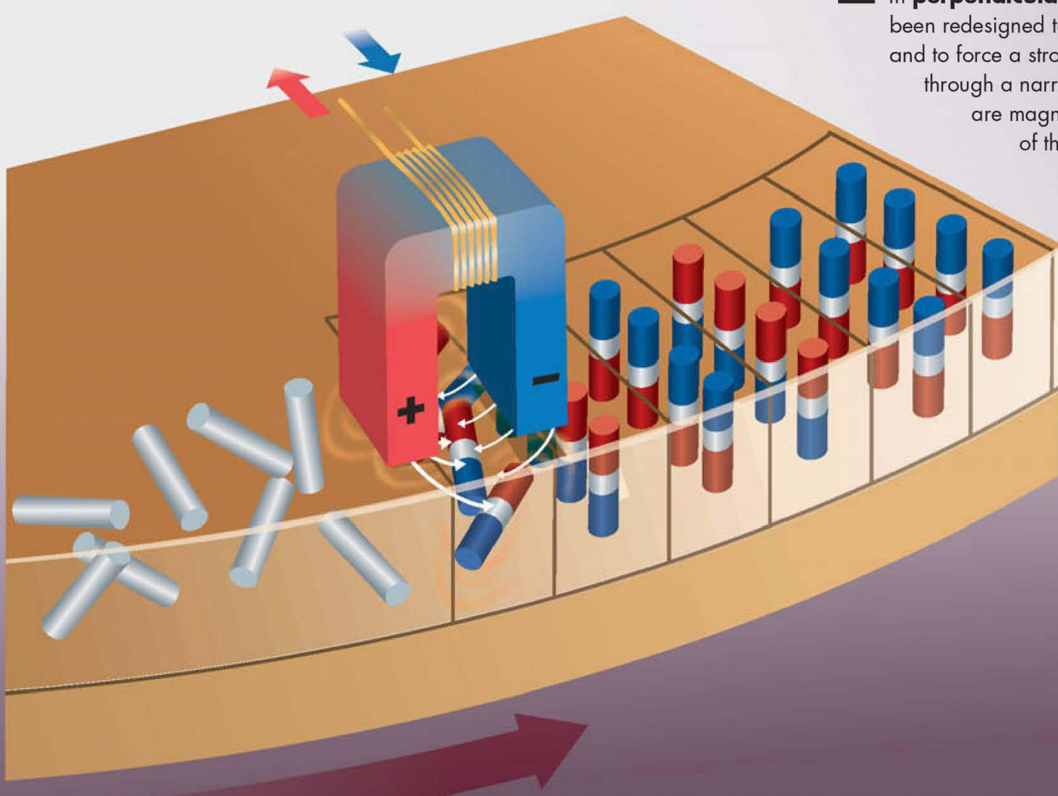
## Perpendicular Recording

**1** Ever since IBM introduced the first floppy disk in 1971, drives have used a technique called **longitudinal recording**. As explained in Chapter 10, "How a Computer's Long-Term Memory Works," current flows through wire wrapped around an iron core, which then generates an electromagnetic field.

**2** By reversing the direction of the current, the drive magnetizes particles in the metallic coating on a disk so the particles' north and south poles are oriented to create combinations representing a 0 or a 1 bit.

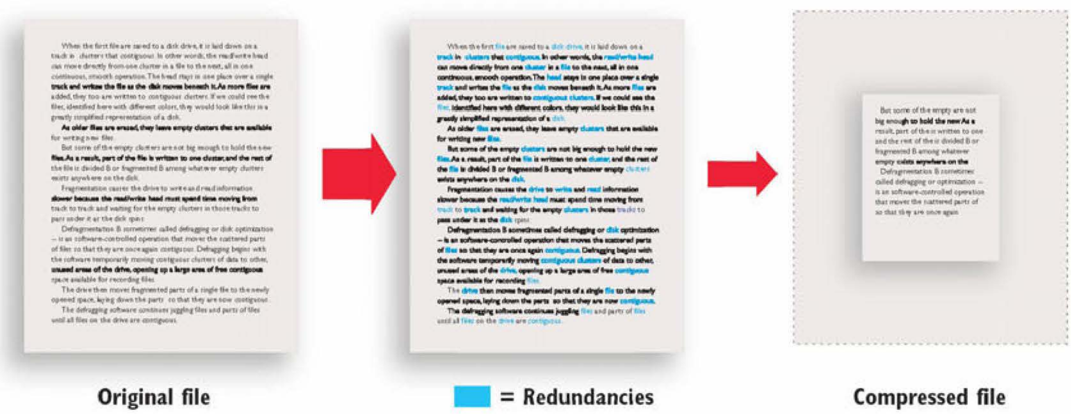
**3** In longitudinal recording, the magnetized particles are aligned with the outer edge of the disk. Attempts to increase storage by making the magnetized areas smaller ran afoul of the **superparamagnetic effect**. The particles' proximity to each other causes their charges to interfere with one another to the point that their north and south poles spontaneously reverse, corrupting data.

**4** In **perpendicular recording**, the iron core has been redesigned to ride close to the disk's surface and to force a stronger electromagnetic field through a narrow area, leaving particles that are magnetized perpendicular to the path of the magnet. In 2007 Hitachi, developer of the drive along with Toshiba, brought to market a 3.5-inch drive using perpendicular recording that stores up to one terabyte of data. That's 1,000 gigabytes.

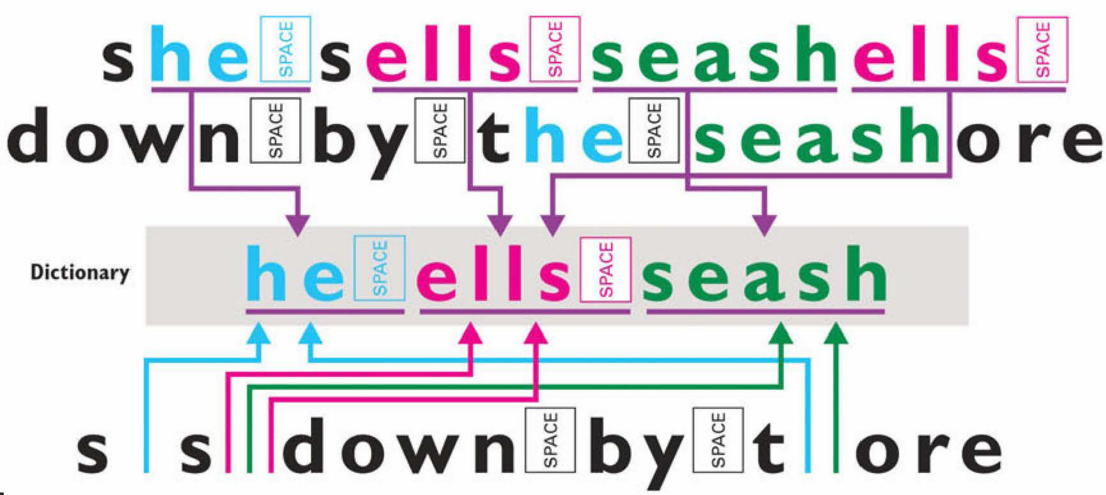


# How File Compression Makes Files Smaller

**1** Files, even those on a compressed disk, can be made still smaller by file compression—a process often called **zipping** a file. This type of compression reduces file size by eliminating the redundancies in the file. This is called **lossless compression**, which means decompression can precisely restore every bit in the file.

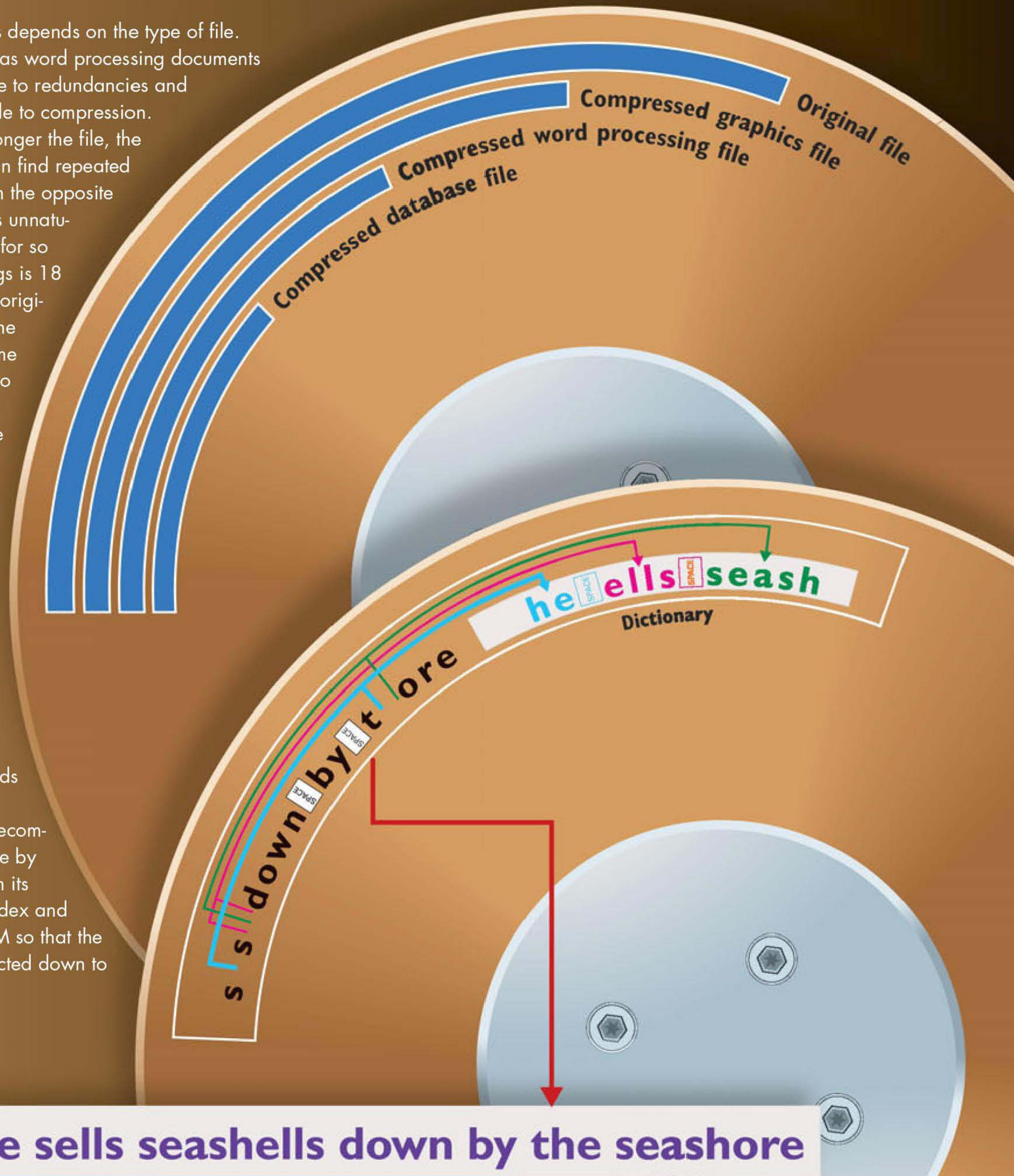



**2** The compression program uses some variation of a scheme generally called *LZ* (after its creators, Lempel and Ziv) **adaptive dictionary-based algorithm**. As the program reads an uncompressed file, it examines the file for recurring patterns of data. When LZ identifies a pattern, instead of writing the pattern in line with the text, as it does other sections of the file, it writes the pattern to a dictionary. The dictionary is stored as part of the compressed file.




**3** Where the pattern would have been written to disk, LZ instead writes a much shorter pointer that tells where the omitted pattern can be found in the dictionary. **Adaptive** means that, as the algorithm writes more of the file, it constantly looks for more efficient data patterns to use and changes the entries in the dictionary on the fly. In the example here, LZ could have chosen "se" in "sells," "seashells," and "seashore" as one of the patterns for the dictionary, but it's more efficient to use "ells" and "seash."

**4** How much the file shrinks depends on the type of file. Some types of files, such as word processing documents and databases, are prone to redundancies and are particularly susceptible to compression. Length matters, too; the longer the file, the more likely it is that LZ can find repeated patterns. In the sample on the opposite page, picked because it's unnaturally full of redundancies for so short a phrase, the savings is 18 percent (40 bytes for the original compared to 33 for the compressed version). Some files can be compressed to as little as 20 percent of their original size. On the average, however, compression is likely to reduce files to only half their original size.

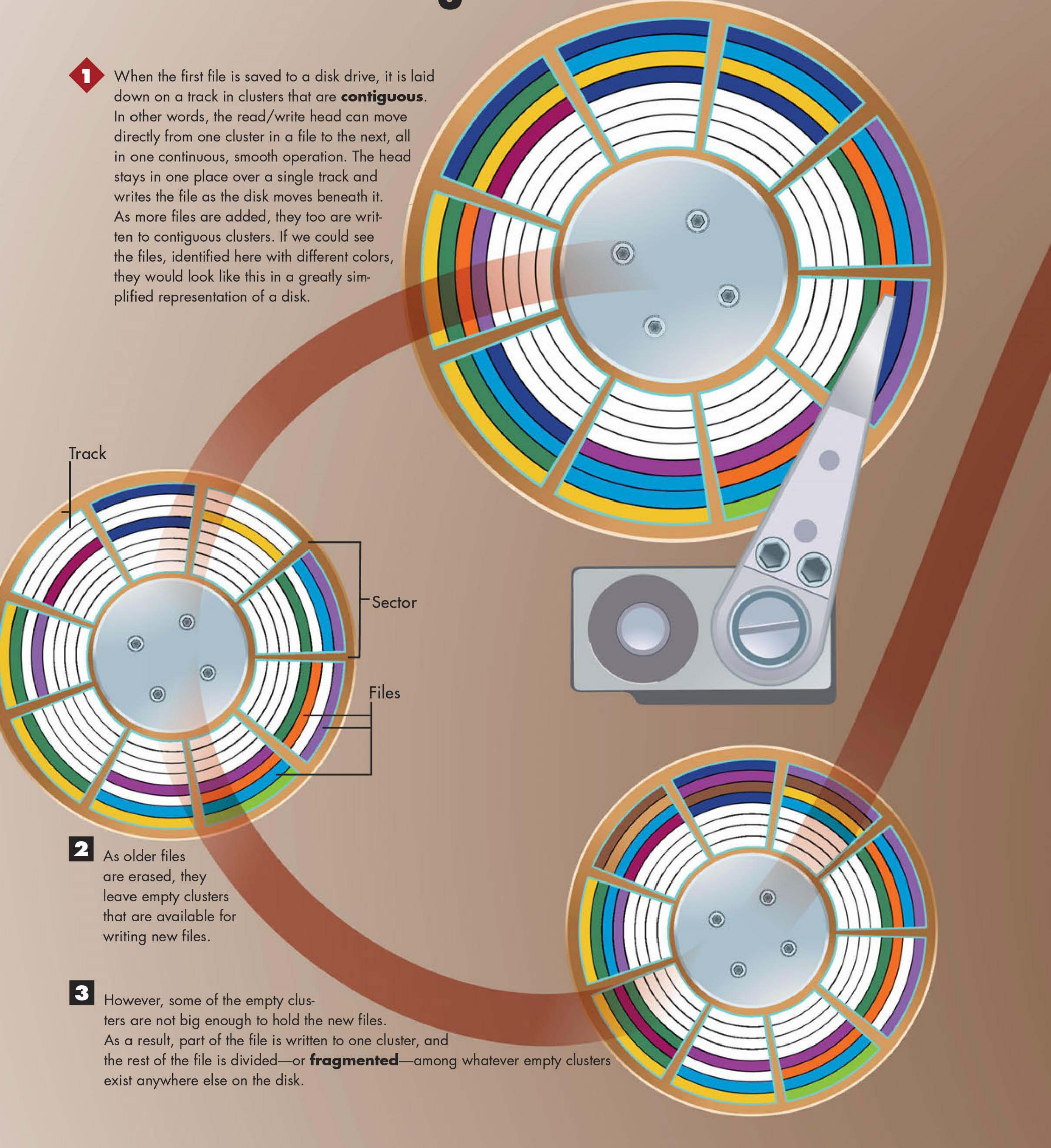



**5** When your computer reads the compressed file and encounters a pointer, it decompresses that part of the file by retrieving the pattern from its place in the dictionary index and writing the pattern to RAM so that the original data is reconstructed down to the last bit.

### Other Compression

The formats for some types of files, particularly graphics and database files, include their own compression. Graphics files often use **lossy compression**, which reduces file size by forever discarding data, such as small variations of color, whose loss won't be noticed. Lossy compressed files cannot be decompressed to their original state. Utility programs, such as PKZip, can further compress files already squeezed by disk compression or built-in file compressions. The utilities use different algorithms that emphasize storage efficiency at the expense of speed.

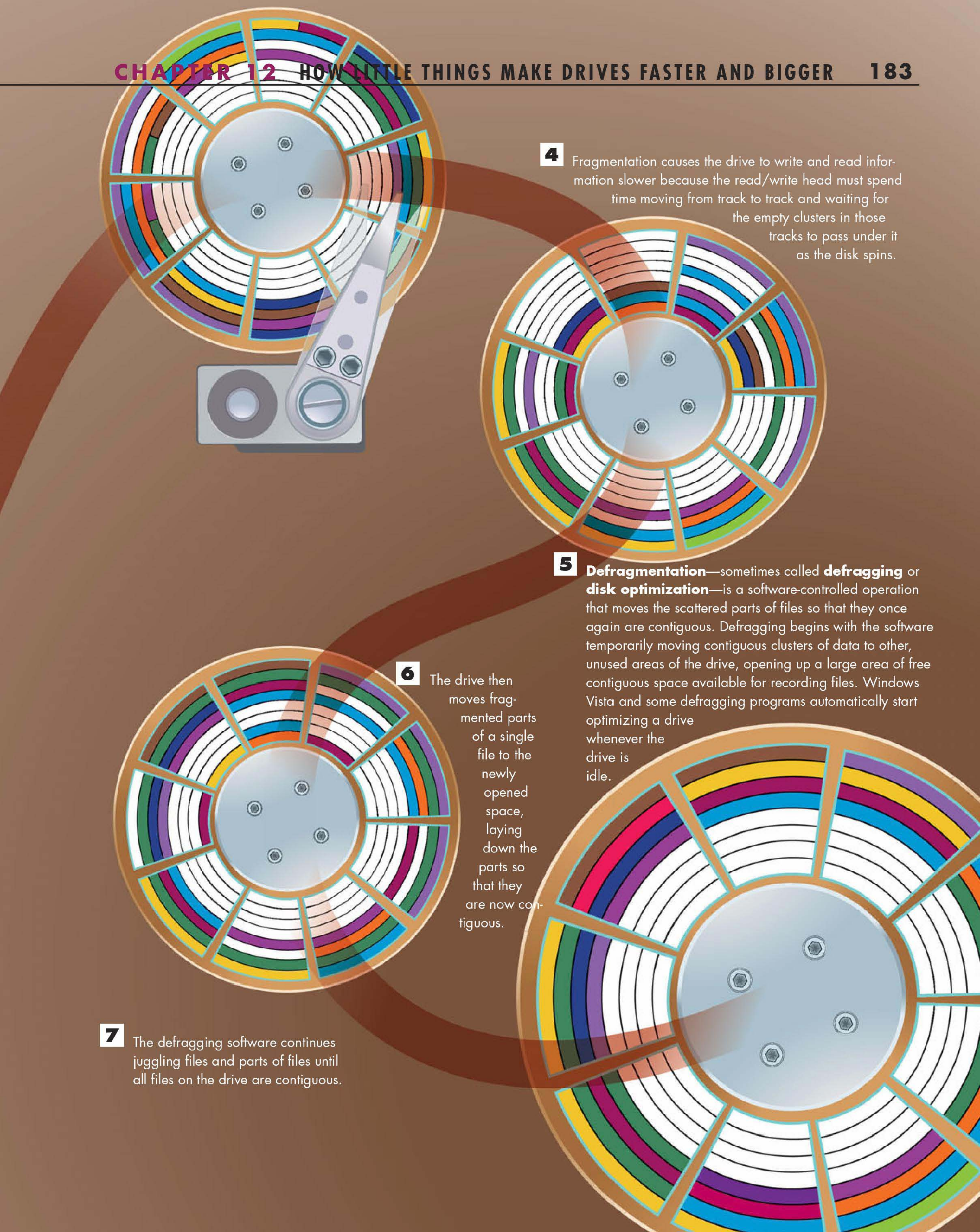
# How Disk Defragmentation Works




**1** When the first file is saved to a disk drive, it is laid down on a track in clusters that are **contiguous**. In other words, the read/write head can move directly from one cluster in a file to the next, all in one continuous, smooth operation. The head stays in one place over a single track and writes the file as the disk moves beneath it. As more files are added, they too are written to contiguous clusters. If we could see the files, identified here with different colors, they would look like this in a greatly simplified representation of a disk.

**2** As older files are erased, they leave empty clusters that are available for writing new files.

**3** However, some of the empty clusters are not big enough to hold the new files. As a result, part of the file is written to one cluster, and the rest of the file is divided—or **fragmented**—among whatever empty clusters exist anywhere else on the disk.



**4** Fragmentation causes the drive to write and read information slower because the read/write head must spend time moving from track to track and waiting for the empty clusters in those tracks to pass under it as the disk spins.

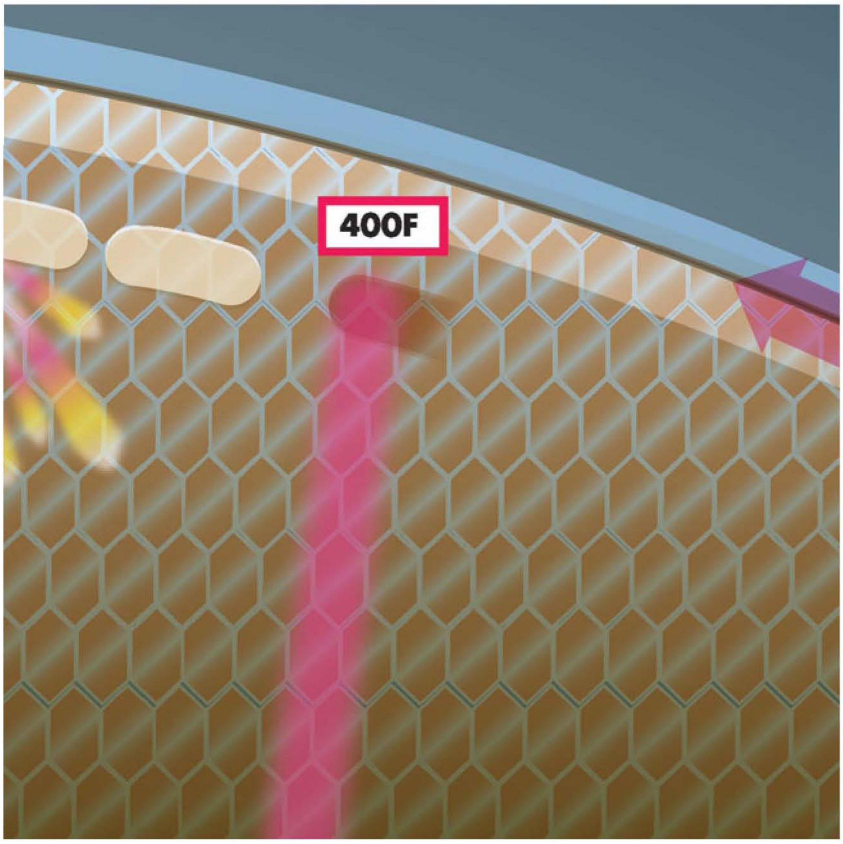
**5** **Defragmentation**—sometimes called **defragging** or **disk optimization**—is a software-controlled operation that moves the scattered parts of files so that they once again are contiguous. Defragging begins with the software temporarily moving contiguous clusters of data to other, unused areas of the drive, opening up a large area of free contiguous space available for recording files. Windows Vista and some defragging programs automatically start optimizing a drive whenever the drive is idle.

**6** The drive then moves fragmented parts of a single file to the newly opened space, laying down the parts so that they are now contiguous.

**7** The defragging software continues juggling files and parts of files until all files on the drive are contiguous.

CHAPTER 13

# How PCs Use Light to Remember Data



**THE** days of magnetic recording are numbered. That number might be very large, but eventually the standard for computer storage will be writing with light in some form of **optical drive**. We're already doing this with recordable and rewritable CDs and DVDs, which use a finely focused laser beam to both write and read data. They sound like the same things, but there's a difference. Data recorded on a CD-**recordable** *(CD-R)* drive cannot be erased or changed. A CD-**rewritable** *(CD-RW)* drive lets you reuse the same disc, much as you use a floppy.

Compared to a floppy disk, either type of disc is gargantuan. Each plastic-encased CD holds up to 700MB of information. The capacity comes from being able to focus a laser beam so tightly that it packs data using a surface area that is smaller than a pinpoint. The relatively clumsy magnetic read/write heads in a conventional drive don't have the precision needed to jam-pack the data. That has made the compact disc the medium of choice for distributing today's software, which takes up hundreds of megabytes. You find CD-ROMs filled with clip art, photographs, encyclopedias, the complete works of Shakespeare, and entire bookshelves of reference material.

As capacious as compact discs are, they're pikers compared to DVD, or **digital versatile discs**—a silly name adopted because the industry wanted to get across the idea that DVDs weren't limited to **digital video discs**, as they were originally called when the only use for them was distributing digitized movies. By either name, DVDs use a two-level storage system that holds at least 8.6 gigabytes of information. And that's using only one side of the disc.

Writable CD and DVD drives are quickly becoming standard equipment on new PCs. The low cost of writable CDs and DVDs makes them a cheap way to back up and transport information as well as a storehouse of MP3 tunes burned to play on your car's music CD player.

As revolutionary as both the CD and DVD have been, they pale in comparison to the potential of two new high-capacity optical disc formats: Blu-Ray and HD-DVD. Both of these formats, which look no different than any other CD or DVD, hold an enormous amount of data. A single layer HD-DVD disc can hold 15GB of data. A single layer Blu-Ray disc holds between 23GB and 27GB of data. And no matter which flavor becomes your preference, both can hold multiple layers of data. No wonder their primary application, to date, has been for holding feature films in high-definition video.

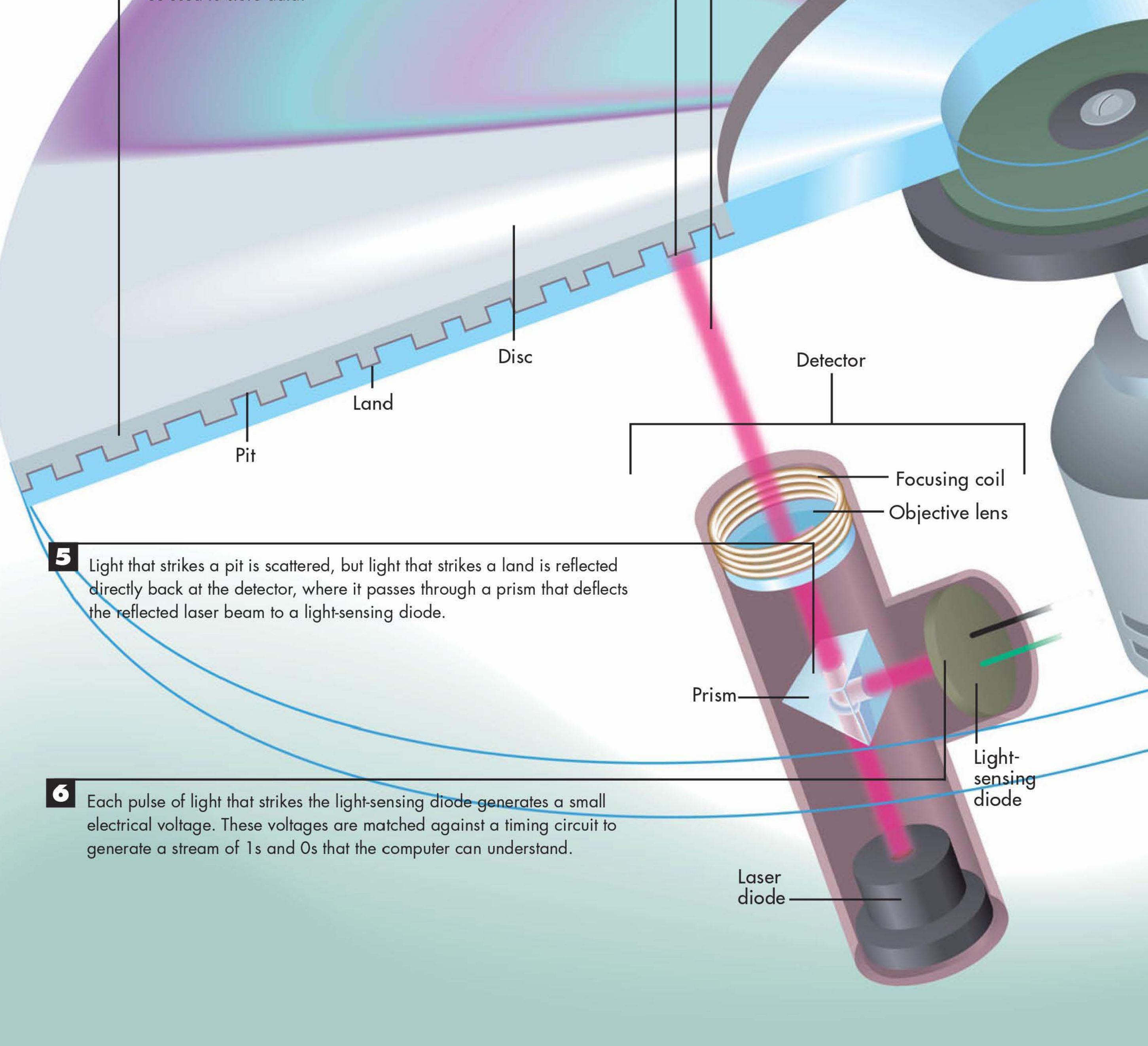
Waiting in the wings are storage methods, such as holograms, that expand the capacity of PC storage into the scale of the **terabyte**—a thousand billion bytes, a thousand gigabytes, or 20,000 four-drawer filing cabinets filled with text.

# How a CD-ROM Drive Works

**2** The laser projects a concentrated beam of light that a lens and a **focusing coil** focus even further.

**3** The laser beam penetrates a protective layer of plastic and strikes a reflective layer that looks like aluminum foil on the bottom of the disc.

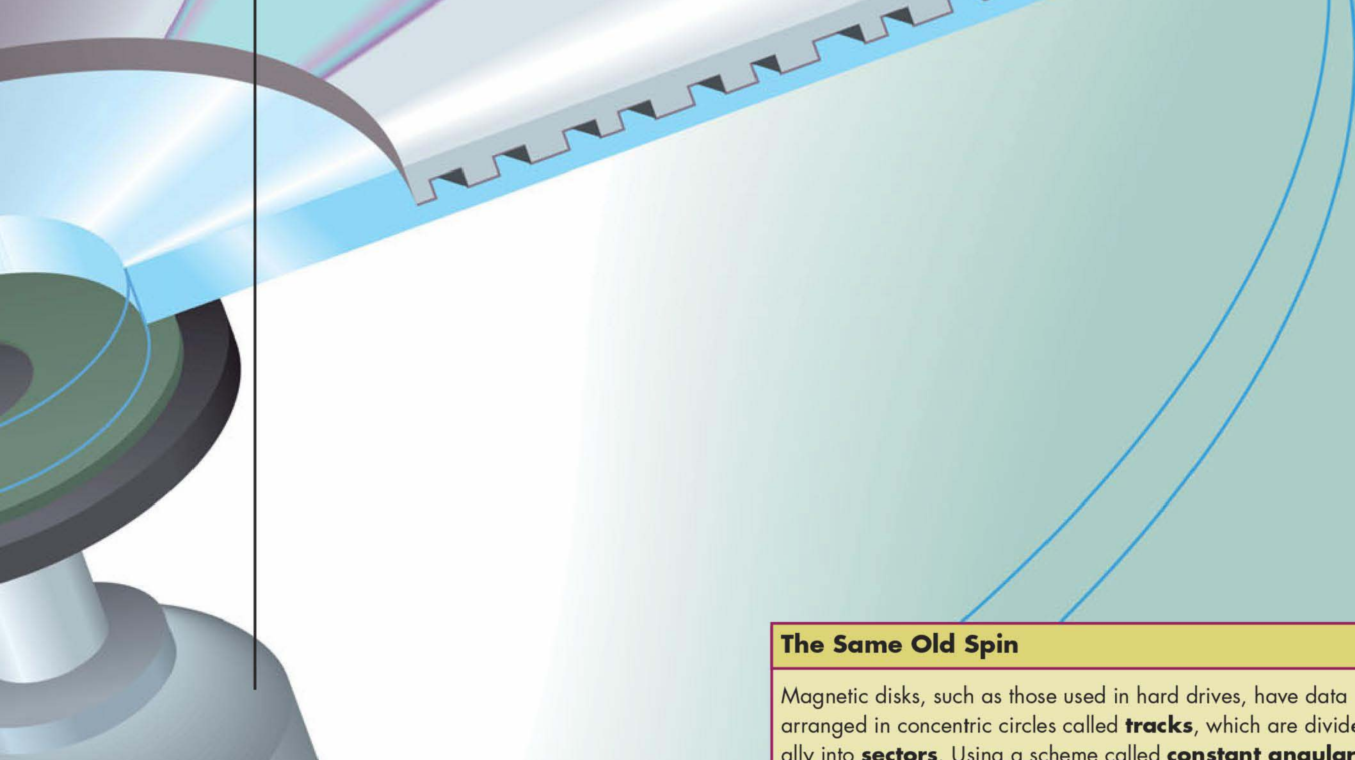
**4** The surface of the reflective layer alternates between lands and pits. **Lands** are flat surface areas; **pits** are tiny depressions in the reflective layer. These two surfaces are a record of the 1s and 0s used to store data.

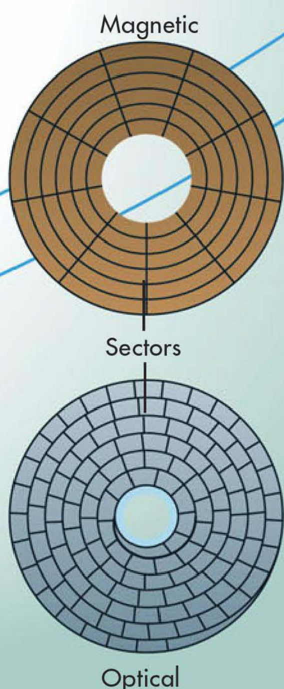



**5** Light that strikes a pit is scattered, but light that strikes a land is reflected directly back at the detector, where it passes through a prism that deflects the reflected laser beam to a light-sensing diode.

**6** Each pulse of light that strikes the light-sensing diode generates a small electrical voltage. These voltages are matched against a timing circuit to generate a stream of 1s and 0s that the computer can understand.

1 A motor constantly varies the rate at which a CD-ROM disc spins so that, regardless of where a component, called a **detector**, is located in relation to the radius of the disc, the portion of the disc immediately above the detector is always moving at the same speed.





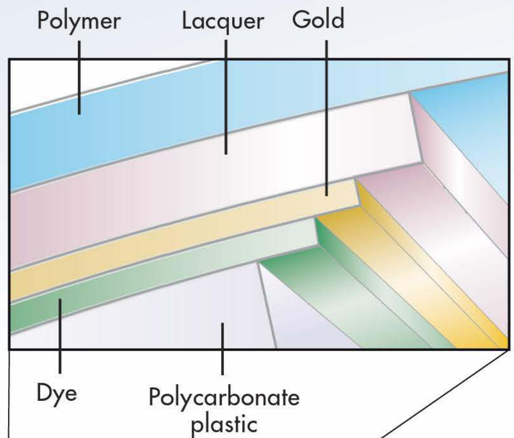

### The Same Old Spin

Magnetic disks, such as those used in hard drives, have data arranged in concentric circles called **tracks**, which are divided radially into **sectors**. Using a scheme called **constant angular velocity**, the magnetic disk always spins at the same rate; that is, the tracks near the periphery of the disk are moving faster than the tracks near the center. Because the outside sectors are moving past the read/write heads faster, the sectors must be physically larger to hold the same amount of data as the inner sectors. This format wastes a great deal of storage space but maximizes the speed with which data can be retrieved.

### A Different Spin

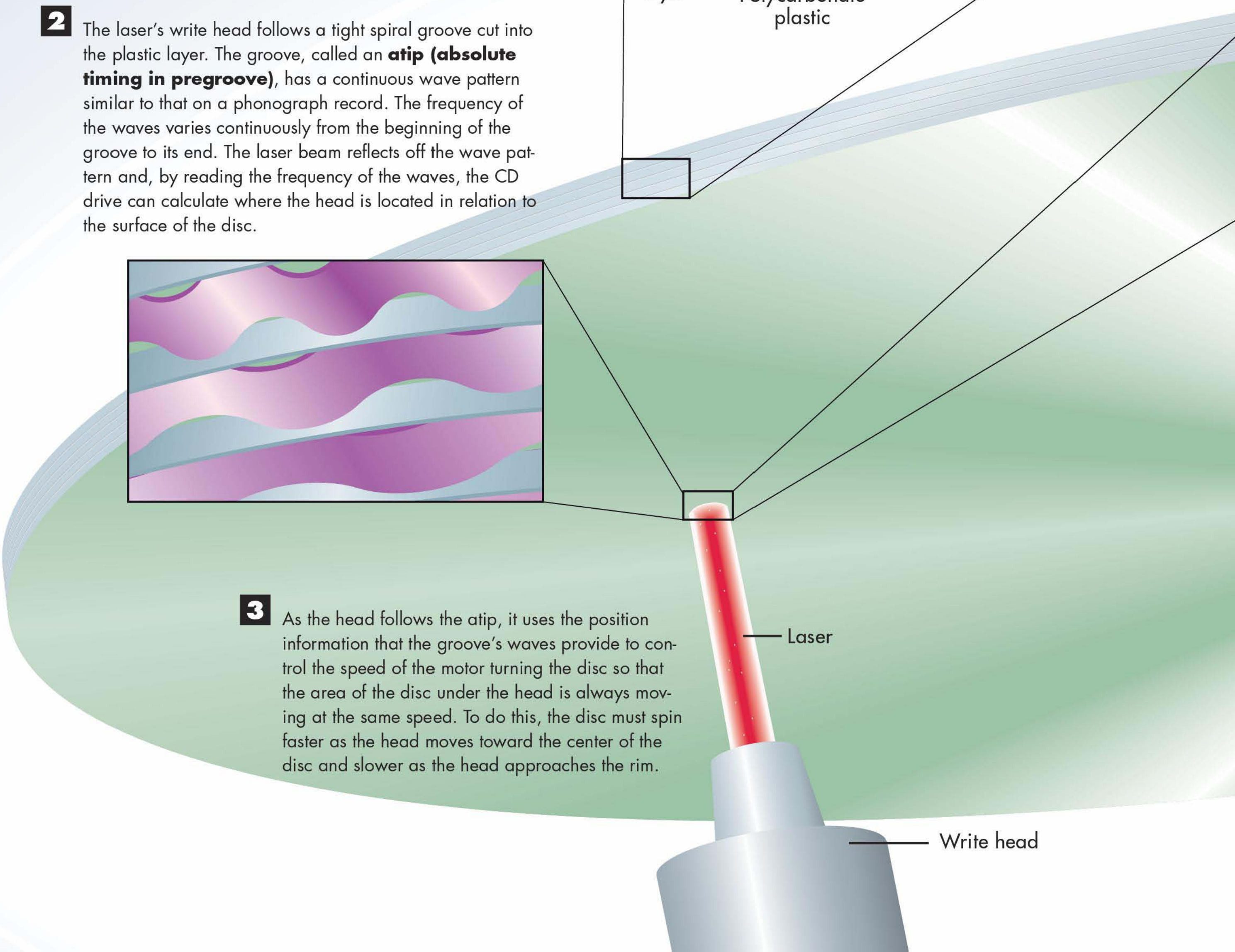
Typically, CD-ROM discs use a different scheme than magnetic disks do to stake out the areas of the disc where data is recorded. Instead of several tracks arranged in concentric circles, on the CD-ROM disc, data is contained in a single track that spirals from the center of the disc to its circumference. The track is still divided by sectors, but each sector is the same physical size. Using a method called **constant linear velocity**, the disc drive constantly varies the rate at which the disc is spinning so that as the detector moves toward the center of the disc, the disc speeds up. The effect is that a compact disc can contain more sectors than a magnetic disk and, therefore, more data.

# How a Recordable CD-ROM (CD-R) Works



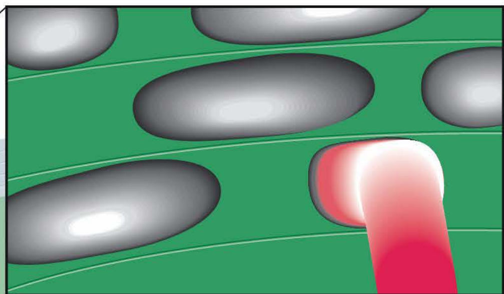

**1** A laser sends a low-energy light beam at a compact disc built on a relatively thick layer of clear polycarbonate plastic. On top of the plastic is a layer of dyed color material that is usually green, a thin layer of gold to reflect the laser beam, a protective layer of lacquer, and often a layer of scratch-resistant polymer material. There might be a paper or silk-screened label on top of all that.

**2** The laser's write head follows a tight spiral groove cut into the plastic layer. The groove, called an **atip (absolute timing in pregroove)**, has a continuous wave pattern similar to that on a phonograph record. The frequency of the waves varies continuously from the beginning of the groove to its end. The laser beam reflects off the wave pattern and, by reading the frequency of the waves, the CD drive can calculate where the head is located in relation to the surface of the disc.



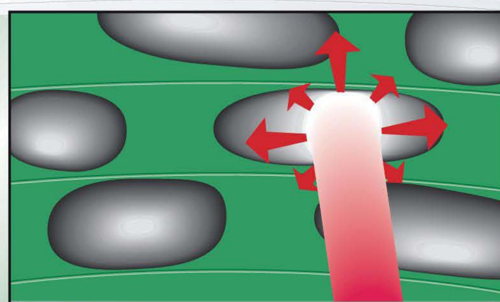

**3** As the head follows the atip, it uses the position information that the groove's waves provide to control the speed of the motor turning the disc so that the area of the disc under the head is always moving at the same speed. To do this, the disc must spin faster as the head moves toward the center of the disc and slower as the head approaches the rim.

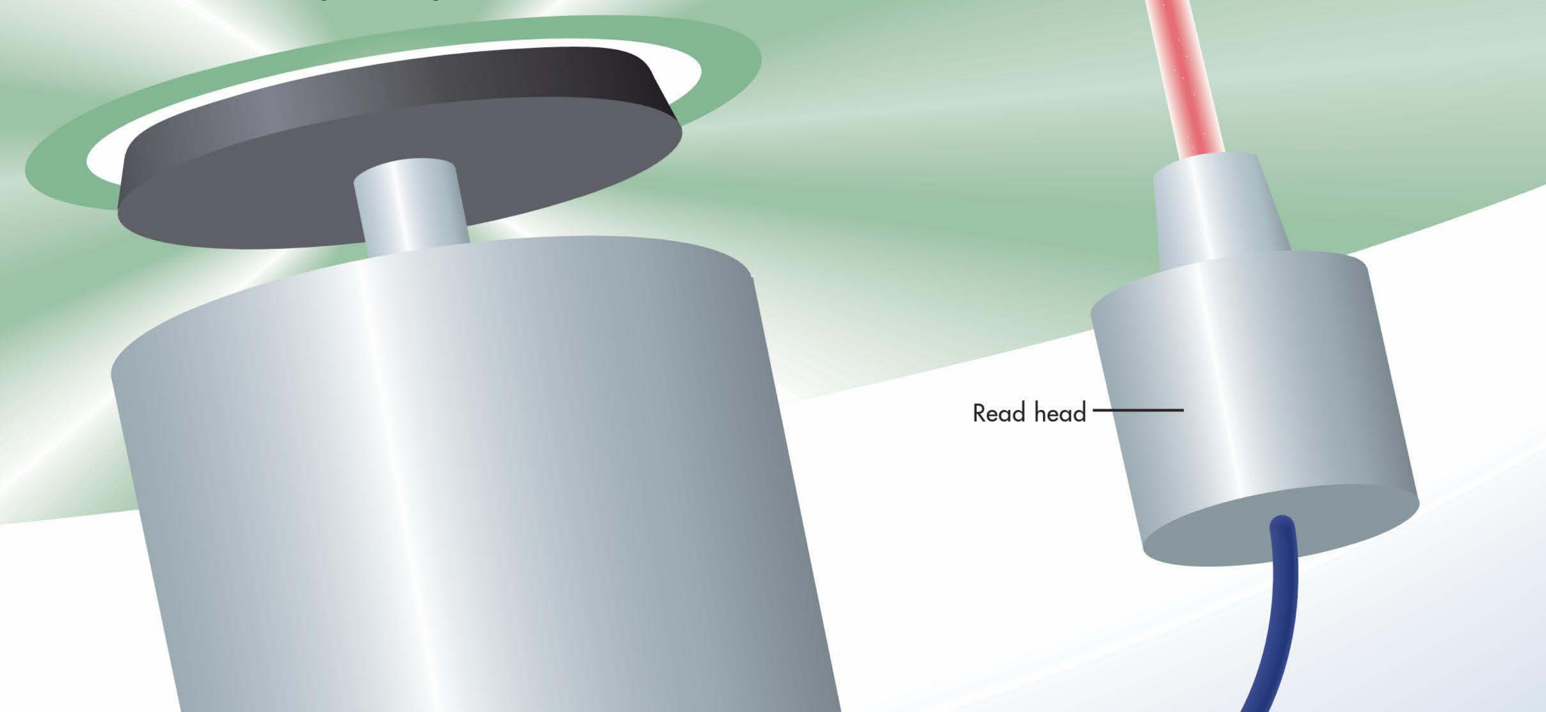
**5** The dye layer is designed to absorb light at that specific frequency. Absorbing the energy from the laser beam creates a mark in one of three ways, depending on the design of the disc. The dye might be bleached; the polycarbonate layer might be distorted; or the dye layer might form a bubble. Regardless of how the mark is created, the result is a distortion, called a **stripe**, along the spiral track. When the beam is switched off, no mark appears. The lengths of the stripes vary, as do the unmarked spaces among them. The CD drive uses the varying lengths to write the information in a special code that compresses the data and checks for errors. The change in the dye is permanent, making the recordable compact disc a write-once, read-many (WORM) medium.



**4** The software used to make a compact disc recording sends the data to be stored to the CD in a specific format, such as ISO 9096, which automatically corrects errors and creates a table of contents. The table is needed because there is nothing like a hard drive's file allocation table to keep track of a file's location. The CD drive records the information by sending a higher-powered pulse of the laser beam at a light wavelength of 780 nanometers.

**6** The CD-Recordable drive—like an ordinary read only CD drive—focuses a lower-powered laser beam onto the disc to read data. Where a mark has not been made on the surface of the disc, the gold layer reflects the beam straight back to the read head. When the beam hits a stripe, the distortion in the groove scatters the beam so that the light is not returned to the read head. The results are the same as if the beam were aimed at the lands and pits in an ordinary CD-ROM. Each time the beam is reflected to the head, the head generates a pulse of electricity. From the pattern in the pulses of current, the drive decompresses the data, error-checks it, and passes it along to the PC in the digital form of zeros and ones.





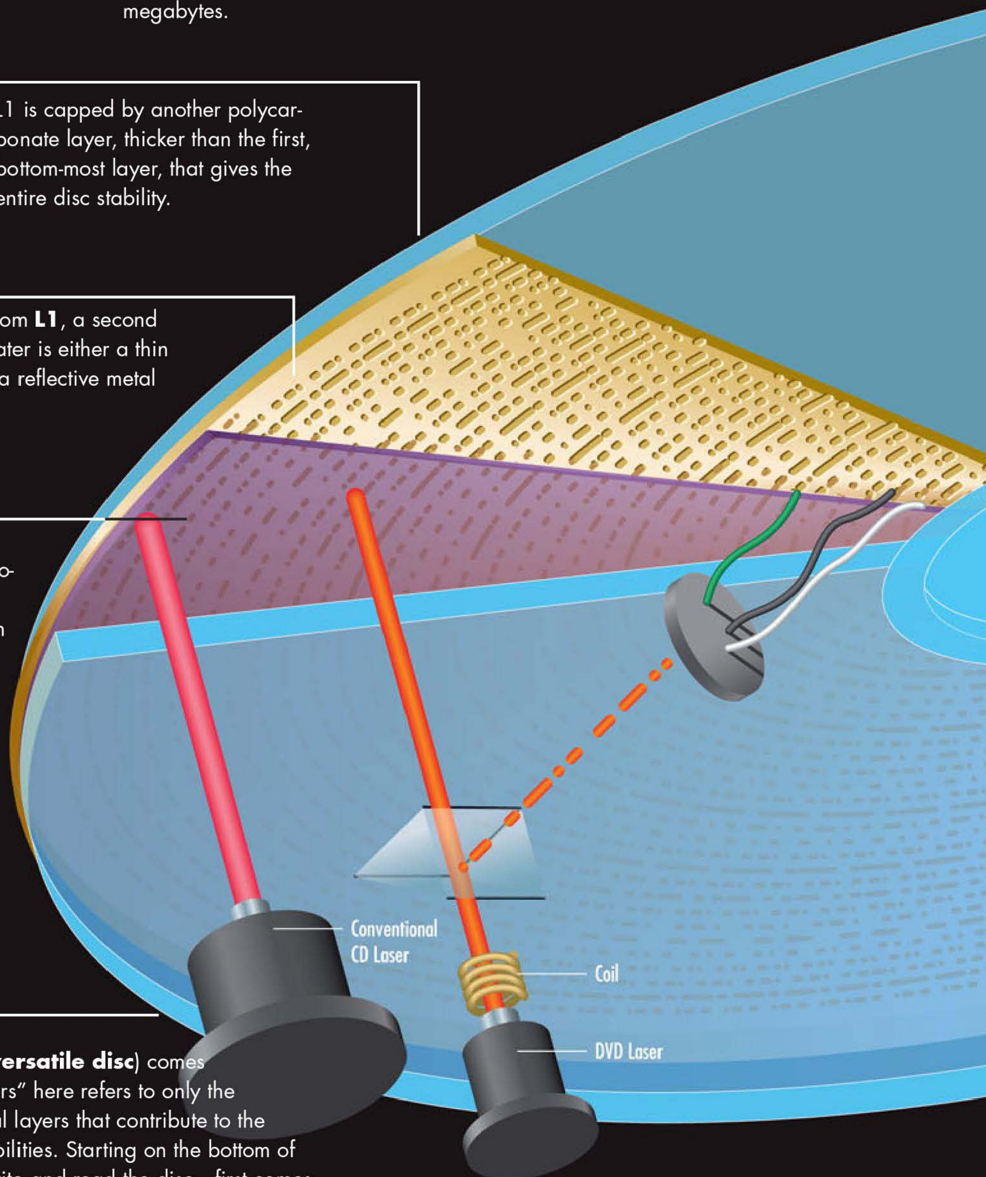
# How a Double-Layer DVD Works

**5** The processes of writing and reading data with a DVD are essentially similar to the methods involving CDs and CD-Rewritable discs in the previous illustrations—with two major differences. One is that the DVD drive uses a red laser with a shorter wavelength than the infrared laser used with CDs to read and write discs. The shorter wavelength creates a narrower beam that, in turn, lets the disc have smaller lands and pits wound in a tighter spiral. This alone lets a DVD hold 4.7 gigabytes of data, seven times as much as a CD's 700 megabytes.

**4** L1 is capped by another polycarbonate layer, thicker than the first, bottom-most layer, that gives the entire disc stability.

**3** Another layer of clear plastic separates L0 from **L1**, a second layer for holding data. Like L0, this second later is either a thin sheet of stamped metal or a dye backed by a reflective metal sheet.

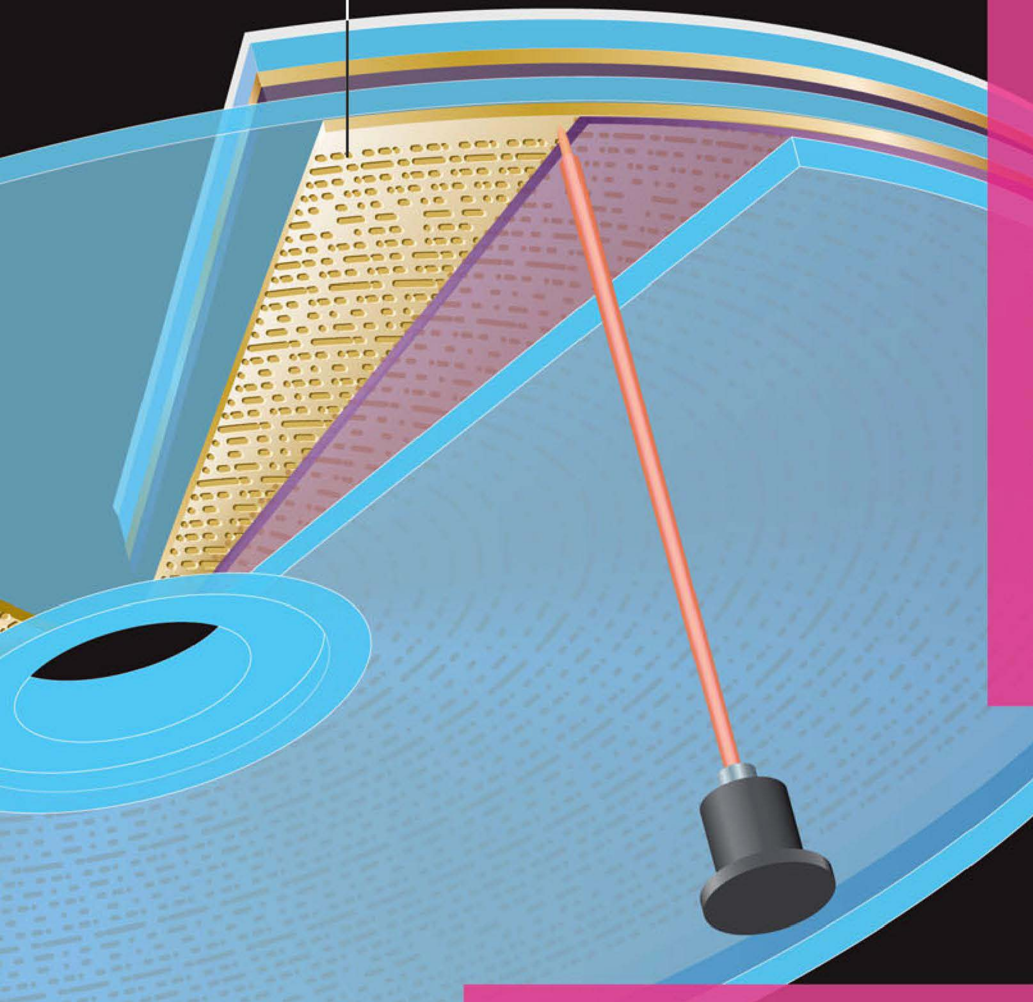
**2** Next comes **L0**, the first of two layers for holding data. If the DVD is commercially produced, this layer has a track of lands and pits created at the factory by pressing a thin sheet of metal with a stamp that is a mirror image of the pits and lands. If the DVD is sold for home use, this layer consists of a dye that responds to the heat from a laser beam. (Lands and pits and their creation are covered in the previous two spreads.) On home-use DVDs, a thin layer of semi-transparent metal reflects the laser beam after it has passed through the dye.

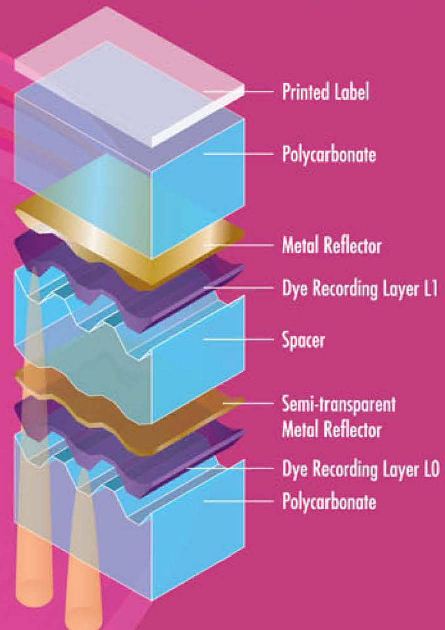


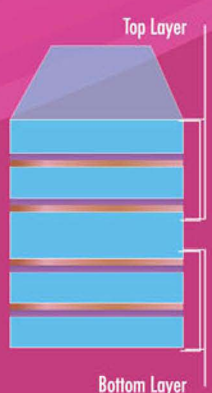

**1** A **DVD** (**digital video disc** or **digital versatile disc**) comes in one-layer and dual-layer versions. "Layers" here refers to only the recording layers. There are actually several layers that contribute to the disc's protection or recording/playback abilities. Starting on the bottom of the DVD—which faces the laser used to write and read the disc—first comes a coat of clear, protective polycarbonate plastic.

**6** The second difference is L1, the second layer of reflective metal or dye, which doubles the capacity of a DVD to 8.5GB. The same laser is used to write or read with the second layer, but just as your eye can change its focal length to concentrate on a near or far object, the laser's lens changes the focus of the beam so that it passes through the first layer of dye/metal harmlessly and strikes L1 with the proper density to read or write pits.

**7** The potential capacity of a DVD doesn't end at the second layer. By applying the same materials to the other side of a disc, its capacity doubles again, to 17GB. But except for some commercial applications, double-sided, dual-layer DVDs are rare and likely to become rarer still since the introduction of high-definition DVDs. (See following spread.)



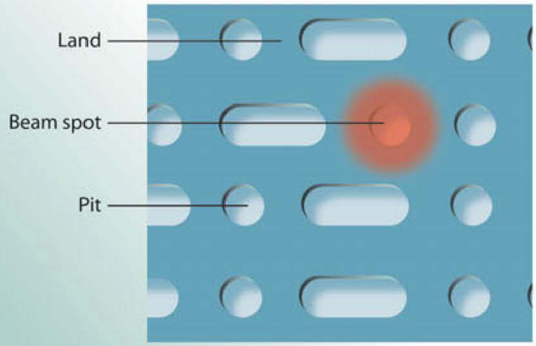


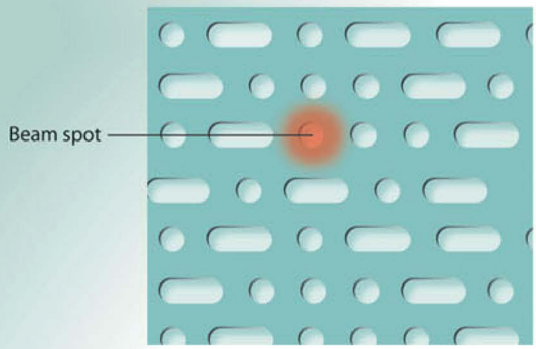

The spiral tracks of data recorded on the two layers of a **double layer DVD** coil in different directions. Starting at the center of the disc, a laser head follows the first spiral track (shown here in blue) until it gets to the outer edge of the disc. Then, with the disc still moving in the same direction, the laser begins tracking the second layer's spiral (red) moving in toward the center of the disc. This design avoids a delay in the flow of data, which is especially important when a DVD contains multimedia.

# How DVDs Play the Blues

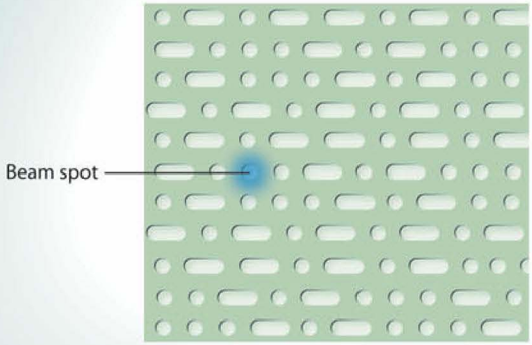
The technology to record movies on DVD in high definition has sprung on the public in the form of a competition. Two competing, incompatible designs to use the ultra-thin beam from a blue light laser are battling for supremacy in much the way Betamax and VHS video tape formats battled it out a couple decades ago. Sony offers a DVD reader (writers will come later) that it calls **Blu-Ray**. Toshiba is selling **HD-DVD**. Both have courted movie studios to convince them to sell high-definition versions of their films exclusively in their formats. Some studios have agreed; other are pressing movies in both formats. The quality of video and audio is similar on both formats. But there is one puzzling difference: The width of the blue laser is what makes it possible to pack onto a normal-size DVD all the additional data high definition demands. The lasers in both DVD drives have the same width—and yet a Blu-Ray disc holds 26GB of data and an HD-DVD disc holds only 15GB. Here are the differences between blue laser DVD and old optical discs and how the two battling blue light differ from each.



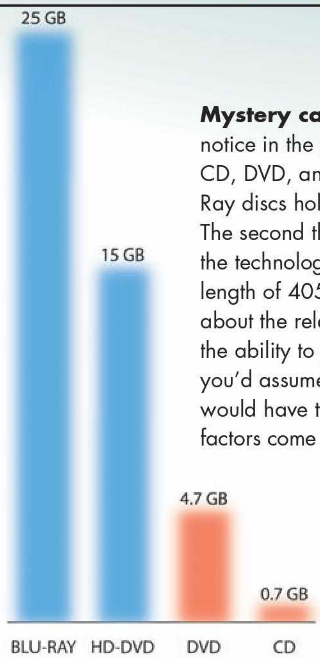

**Compact Disc:** A CD uses a red laser that has a wavelength of 790 nanometers (nm). That's a little less than a millionth of a meter; it would take more than 300 dots of red laser light to cover the width of a human hair. The distance between coils of the spiral track carrying data is 1.6 microns (1.6 millionths of a meter). The data is encoded in the form of **pits** (holes) and **lands** (unchanged, level areas).




**DVD**: The common DVD that's been around for about 10 years also uses a red laser beam, but it is narrower—650nm. That small difference allows a single-sided, single-layer disc to hold seven times as much as a CD. The coils of the tracks are 0.7 microns apart and the lands and pits are smaller.

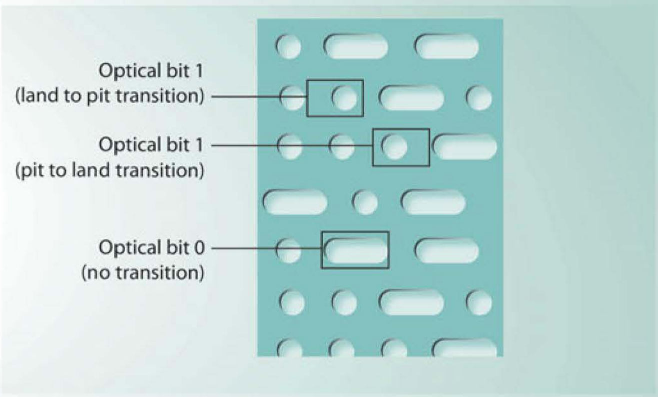



**HD-DVD** and **Blu-Ray**: With these high-definition formats, optical technology takes a leap to much smaller pit and land sizes. Both technologies use a blue-purple laser with a beam that's 405nm wide. The coils of their tracks are 0.3-0.4 microns apart, about a fourth of the distance that separates common circuit traces in microprocessors.



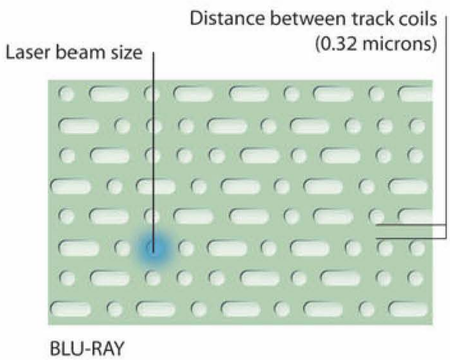

**Mystery capacities:** The first thing you might notice in the graph showing the data capacities of CD, DVD, and blue laser optical discs is that Blu-Ray discs hold nearly twice as much as HD-DVD. The second thing you might notice is that each of the technologies uses a laser beam with a wavelength of 405nm. Based on what we've learned about the relationship between wavelength and the ability to pack more and more bits onto a disc, you'd assume the two competing blue laser drives would have the same data capacities. A couple of factors come between us and an easy explanation.

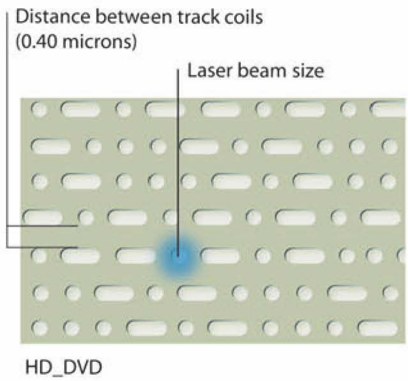
**Optical bits:** One reason is the way bits are recorded and interpreted. Optical discs actually use the transition between lands and pits and pits and lands to represent a 1 bit and the lack of such a transition to be a 0 bit. But these are **optical bits**, not the bits that are used to record music and videos on DVDs.




**Data bits:** Before optical bits become data bits, the optical drives must demodulate them, a process that takes several of the optical bits and converts them into single 0 and 1 data bits. HD-DVD uses a process called **twelve to eight demodulation**. It converts each group of 12 optical bits into 8 data bits.

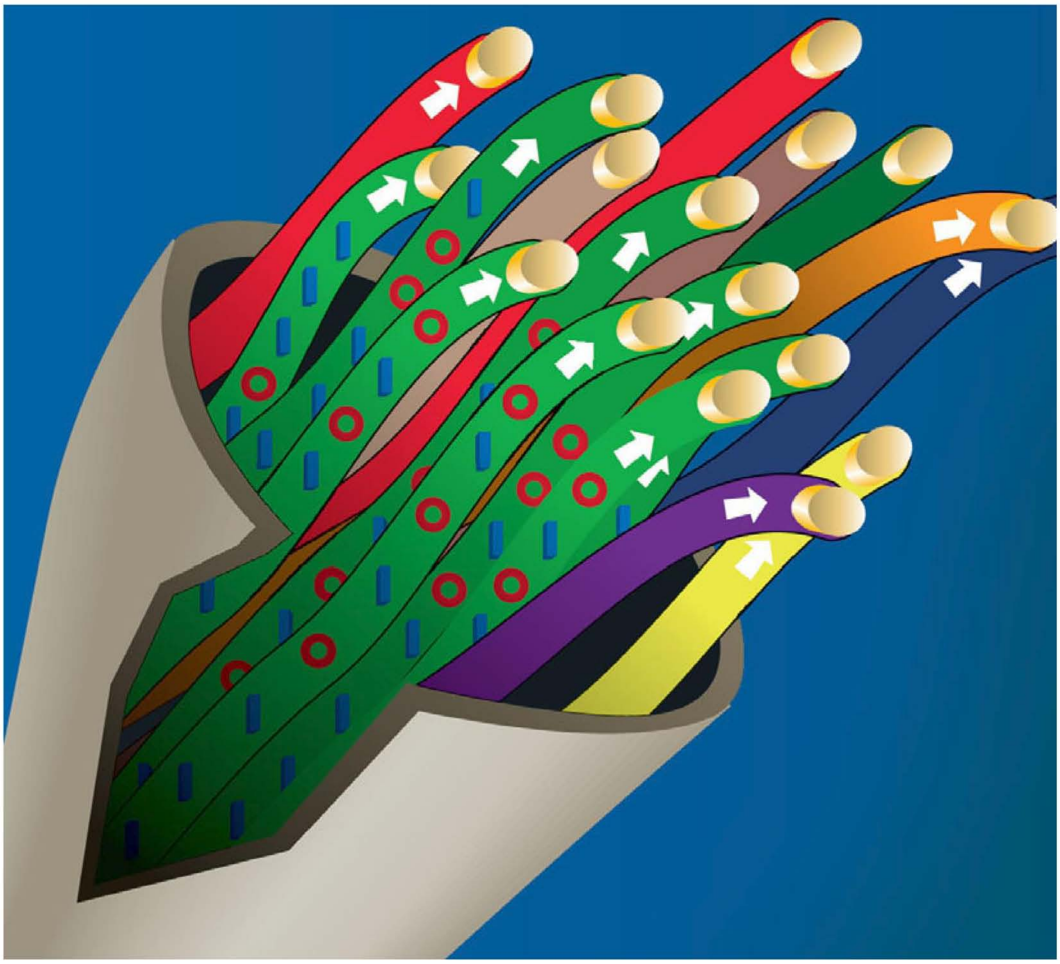
**Protection versus capacity:** The big difference comes in the protective plastic layer on the two types of DVDs. The Blu-Ray layer is only 0.1mm thick, which allows the disc's higher recording density. HD-DVD uses a thicker layer (0.6mm) that provides more protection but which forces the drive to use wider lands and pits.




## Blue Movies

In a similar technology battle—between Sony's Betamax video tape recorders and VHS sponsored by a consortium of electronics makers—technology was not the deciding factor that eventually led to VHS's triumph over Betamax, which was considered better technically and in the quality of recordings. The reasons VHS won out are that it was cheaper to make tapes for it, and the biggest source of video was porno, an industry that doesn't waste money on frills such as resolution, audio, and clothes. Of Blu-Ray and HD-DVD, the latter discs are cheaper to produce, leading industry observers to predict HD-DVD is the likely winner after porn purveyors make it the blue laser disc of choice.



**30,000 B.C.**
Paleolithic people in central Europe record numbers by notching up tallies on animal bones, ivory, and stone.

**500 B.C.**
Oldest known objects used to represent numbers, bones with notches, are discovered in western Europe. A wolf bone more than 20,000 years old with 55 notches in groups of five is discovered in Vestonice, Czechoslovakia in 1937.

**1863**
Giovanni Caselli receives U.S. patent for a fax machine called the "pantelegraph" based on Alexander Bain's 1840 idea of synchronized pendulums. Service between Paris and Lyons, France begins between 1865–1870, ending with the Franco-Prussian War.

**1867**
Christopher Sholes, a Milwaukee newspaper editor, invents the typewriter. Six years later, E. Remington & Sons of New York refines and markets Sholes' design.

**1878**
The first shift-key typewriter appears.

**1884**
Nipkow (Germany) devises scanner for scanning and transmitting images.

**1887**
Bell and Tainter organize the American Graphophone Co. on May 13, financed by court and congressional reporters James Clephane and Andrew Devine and John H. White, to make and sell the treadle-powered graphophone as a dictation device for businesses.

**1917**
Teletypewriters appear, allowing point-to-point printed communications. Today's TTY terminal emulation standard dates back to this device.

P A R T

# 5

# Input/Output Devices

C H A P T E R S



**1925**
AT&T's Long Lines Department offers the press an early facsimile service between New York, Chicago, and San Francisco.

**1936**
A keyboard designed for the ease of the user is developed by John Dvorak. The keyboard is designed with the least used keys on the outside corners, and the most often used keys within easy reach of the user's fingers.

**1948**
Britain's Manchester Mark I is the first computer that can store a program electronically rather than requiring programmers to input data by setting switches manually.

**1949**
EDSAC (Electronic Delay Storage Automatic Calculator), built at the University of Cambridge, is capable of translating alphabetic shorthand into the binary it needs to understand the problems posed.

**1950**
Douglas Engelbart realizes screens could be used instead of paper to view computer output.

**1952**
G. W. Dummer, a radar expert from Britain's Royal Radar Establishment, presents a paper proposing that a solid block of materials be used to connect electronic components, with no connecting wires.

**1956**
At MIT, researchers begin experimentation on direct keyboard input on computers, a precursor to today's normal mode of operation.

**1960**
Fiber-tip pen is introduced.

**1962**
Teletype ships its Model 33 keyboard and punched-tape terminal, used for input and output by many early microcomputers.

*GIGO*

—Acronym for "Garbage in, garbage out," the programmer's way of saying that the results you get out of a computer are no better than the data you feed into it.

**REAR** Admiral Grace Hopper was an early computer engineer who helped create the language COBOL and invented the term "bug" when she literally found a dead insect that had caused Harvard's Mark II computer to malfunction. Dr. Hopper often taught computing classes, where she would ask, "What is a microsecond?" To answer her own question, Dr. Hopper pulled several wires out of her handbag. Each wire was about 11 inches long, which she had calculated was the distance an electron travels in one millionth of a second.

We don't usually think of distance being an issue in the everyday appliances and electronics we use, including our computers. I know I tend to think of electricity as instantaneous. After all, moving at 186,000 miles a second, what's a few inches here or there? But on the scale where electrons move and become bits of information, it's not just seconds that count, or even microseconds. In microchips and circuits, things happen in a matter of **nanoseconds**—billionths of a second. Chips can contain as much as a quarter of a mile of wire traces through which millions of pulses of electricity are being pushed every second. The territory electricity has to cover is crucial enough that Cray supercomputers were built in a circle to minimize the distance electrical signals traveled.

As if requirements that signals move at light speed weren't demanding enough, the signals must share the traces of aluminum and copper on circuit boards and in chips. Timing is everything. One computer component's output is another component's input. A file being read from disk is output as far as the drive is concerned, but for the memory chips receiving the file, the same data is input. If hard drives are slower than memory chips—and they are—that means that memory is left cooling its heels, accomplishing nothing, while the hard drive grinds out data bits. Much of the design of a PC is aimed at pipelining input and output to eliminate awkward delays in a lightning-fast ballet of bits.

Among the foremost timing masters are caches. A **cache** is any storage—RAM, a microprocessor's own memory, or hard drive—that's used as a buffer to contain frequently used data that would otherwise come from a much slower source. Speed is relative. Memory chips cache

**1962**
Ivan Sutherland creates Sketchpad, a drawing program that uses a light pen, allowing easy manipulation of graphics and text onscreen.

**1963**
Finalization of the ASCII code (American Standard Code for Information Interchange) permits machines from different manufacturers to exchange data.

**1963**
Douglas Engelbart receives a patent on the mouse pointing device for computers.

**1964**
IBM announces System 360, a family of computers that can be used for science and business and share the same software, printer, and tape drives.

**1964**
Touchtone phone is introduced.

**1968**
Rand Laboratories develops the Rand Tablet, which would translate handwriting into typed text.

**1968**
Doug Engelbart of the Pentagon's Advanced Research Projects Agency (ARPA) demonstrates a new way of interacting with a computer using a keyboard, mouse, and a graphic rather than a text-based interface. The mouse is a block of wood with a single button.

**1969**
Barcode scanner is invented.

**1969**
The RS232-C standard for communication permits computers and peripheral devices to transmit information serially—that is, one bit at a time.

data from slower hard drives. But caches on hard drives hold copies of information that, if they weren't on the drive, would have to be drawn from the still slower Internet. A delay for one signal can create bottlenecks for thousands of other signals. If two signals collide, trying to use the same circuits at the same time, the result can be fatal. But faster and faster computers require more nimble switches and transistors, able to squeeze through just a few more bits of electricity each billionth of a second.

Electricity does not flow freely. Even conductive materials, such as copper, have electrical resistance built into them. Resistance is like a clogged water pipe. If the pipe's filled with gravel, less water gets through. Impurities in conductors slow the flow of electrons. Small pipes conduct less water than big pipes. The resistance of a conductor increases as the diameter of a wire or the width of an electrical trace decreases. Water pipes, though, don't have to contend with all the distractions that plague electricity. Imagine if water in a pipe had to switch direction thousands of times a second, or that an invisible force emanated from each pipe and interfered with the flow in other pipes. The latter is a phenomenon of **electromagnetism**—the fields of radio waves that are produced by moving electrons.

An early keyboard.

Electromagnetic fields were first discovered in the 19th century, when physicists noticed that **arcing**—leaping sparks of electricity—were replicated from one device to another with no wires connecting them. This led scientists to believe that it was possible to communicate over long distances without wires. The first radio transmitters used electric arcs, ushering in the first wave of what would later become multimedia. Radio in the early 20th century was as exciting and important as the Internet at the end of the century. In other machines, from TV to microwave ovens and radar, we've used electromagnetism to our advantage. The downside is that every device in your computer—for that

**1970**
Xerox's Palo Alto Research Center (PARC) investigates the "architecture of information" and how to make computers easy enough for anyone to use. Using the ideas that people would respond better to intuitive command structures and they don't need to understand how the hardware functions to use the technology, PARC comes up with black-on-white screens, a bitmapped display, icons, pointers, laser printers, word processors, and networks (notably Ethernet). The Xerox Star and the Alto are two computers that embody all these groundbreaking ideas but they are never successfully marketed.

**1971**
Liquid crystal display debuts.

**1974**
Post-It note pads appear.

**1975**
The January edition of *Popular Electronics* features the Altair 8800 computer kit, based on Intel's 8080 microprocessor, on its cover. The machine comes with an open 100-line bus structure that evolves into the S-100 standard.

**1982**
A prototype Macintosh arrives at Microsoft to aid in development of its applications.

**1983**
Microsoft introduces the Microsoft Mouse, a low-cost, hand-held pointing device for use with the IBM PC, as well as any MSDOS–based personal computer.

**1988**
Doppler radar is introduced.

**1988**
Microsoft sells its one-millionth mouse.

**1988**
Engineers introduce technology to input data by writing onscreen.

**1989**
Global positioning system by satellite is launched.

matter, everything from the phone to the air-conditioning—is spewing out electromagnetic fields of different strengths and frequencies. Those fields, in turn, can affect the flow of electrons in other wires. Often, electromagnetic waves pass through each other invisibly, as different light waves do. But if the frequency or wavelengths of the fields are too similar, they can interfere and distort each other. Then it's called electrical noise, and many of the improvements in computer hardware are related directly to creating new, better sets of electrical ear muffs. In many cables, there are twice as many wires as are needed to carry data. The extra wires are included to absorb noise that might interfere with the working wires.

A prototype of the first mouse.

Noise is important because the nanoworld of electricity is still finite; a voltage doesn't just instantly appear on a wire. It takes time, like turning open a pipe valve. The longer it takes for the valve to open, the fewer times it can be turned on and off in a second, which means fewer chances for drops of data to flow. On the input end, how much voltage is flowing out is important. A trickle is ignored, but finally the voltage rises enough to register itself with the receiving device. The more noise computer components have to contend with, the more difficult it is for a component to tell whether some passing spike in current is intended to convey data or is just stray noise. Many of the improvements in speed and versatility for the first two decades of the personal computer have come incrementally—a few nanoseconds off the time it takes to recognize a change in voltage there, a microsecond shaved off a switching operation here.

We don't see computer operations on this scale, of course. We measure time in the seconds it takes a menu to drop down or the minutes it takes a dial-up modem to negotiate an Internet connection. So, when we talk about input and output, we usually aren't thinking about the subatomic voyages of electrons. We are used to thinking about input and output on a human scale, and that means keyboards, mice, game pads, displays, digital cameras, scanners, printers, and a host of other devices designed to let us get information into a computer and get it out again.

## Human-Computer and Real World Interaction

All the marvelous tasks that a personal computer is capable of doing would be meaningless without some way for the PC to communicate with the world outside itself. The first personal computers, such as the Altair, used a method of communicating so primitive that it's a wonder computing pioneers had the imagination to conceive that these contraptions could be practical in the real world. Program instructions and data were fed into the computer by flipping electrical switches—not miniaturized switches in the form of transistors, but ordinary thumb-sized switches. The results of a computation were presented in the form of a pattern of tiny light bulbs lit on a panel. To the uninitiated, the pattern of lights was incomprehensible.

Altair hobbyists, of course, were quick to come up with a keyboard and a monitor to make communications easier with the new microcomputer. For years, though, the keyboard was computing's main input device, and the monitor or printer were the prime ways to get data out of a computer. But even as keyboards and monitors gave us a way to communicate with a computer, they also tied us down to the computer. Only with the emergence of ultra-portable notebooks less than three pounds has the PC become small enough and light enough to be carried around easily. Wireless networking and a fully charged battery mean that, today, you can compute anywhere within range of your network, totally unencumbered by wires connected to anything.

Most of the advances in input/output, especially in the last decade of the 20th century, have been devoted to making both input and output more natural. The first target has been the keyboard, which makes sense because it's the only part of the PC with which any computer user actually comes into contact with. The first attempted change was a different layout for the keys devised by John Dvorak (not the computer columnist) in 1936. It put the least-used keys along the outside of the keyboard where the weaker pinky finger and ring finger could reach them. The more often used keys were put under the index and middle fingers to reduce the time it takes to move to them. It's certainly a better concept than the current QWERTY key arrangement, which no one can explain, but which all keyboard users, even hunt-and-peckers, are invested in. But the Dvorak keyboard also reveals a fundamental truth about computer technology: The better idea doesn't necessarily win. Of more importance than scientific, mechanical, or electrical excellence is compatibility with past technology. Repeatedly, a better idea has been defeated in the marketplace because it requires too much effort, money, and time to change to different hardware, learn new software, or simply break old habits.


Combination microphone and mouse.

There have been scores of other innovations for keyboards. The best known is the design that splits the keyboard down the middle so the keys lie at more natural angles to how the arms and wrists bend. But there's also been a design that took that approach even further—tilting the two halves of the keyboard up so they become two vertical surfaces controlled by hands with the palms turned toward each other. Relax in a chair with armrests, and you'll see that your hands indeed do come to rest naturally with the palms facing each other. But the totally split keyboard went a step too far; The arrangement is too alien. Some keyboards have been stranger still, designed with the keys in two circular depressions, or as the device from an alternate universe: a one-handed device using chording—pressing different combinations of keys at the same time to represent different letters.

Some of the most accepted input/output innovations have been born at Xerox's Palo Alto Research Center (PARC), which since the 1970s has been studying how people communicate information so as to make computers easy enough for anyone to use, including people who have no idea of how the underlying technology works. PARC created black-on-white screens, a bitmapped display, icons, pointers, laser printers, word

processors, and networks, most notably Ethernet. All these innovations are commonplace today, but a Xerox personal computer is not. It's proof once again that better technology doesn't guarantee a win. Xerox did produce two computers, the Xerox Star and the Alto, that brought all these ground-breaking inventions to realization. Maybe it was too soon, but in any case, they didn't sell.

The input innovation that's received the most acceptance is the mouse combined with a graphic interface. Mouse innovations, if anything, have been stranger than keyboards. I've used pointing devices that looked like ballpoint pens, were worn as a ring, and controlled with your feet. They didn't sell either—which is amazing when you consider there's nothing particularly intuitive about using a mouse. It requires a lot of tricky eye-hand coordination—more than you need when touch typing.

Language is not really intuitive either, but it's an increasingly important part of computer input/output. By the end of the 20th century it really was possible to speak to a computer and have it respond correctly, even speak back to you. It's *possible*, but not quite there. In the next chapter, we'll look at how speech recognition is likely to pan out. Meanwhile, we'll be looking at more down-to-earth forms of input and output, including new devices that eliminate human beings as input devices entirely.

## KEY CONCEPTS

**accelerated graphics port** An expansion slot that gives a video card fast access to bitmap stores in a PC's main RAM. This slot has quickly lost ground to the PCI-Express slot.

**adapter (expansion card)** A circuit board, often with its own microprocessor and memory, that is inserted into an expansion slot to add to a PC's capabilities. For example, video, sound and SCSI cards.

**analog/digital** An analog signal is a continuous, varying electrical output, such as those created by microphones and sound amplifiers. A digital signal consists of discreet, separate values for input or output data.

**analog-to-digital converter (ADC)** A chip that converts an analog signal to digital values that a PC can manipulate.

**ASCII** Acronym for American Standard Code for Information Interchange. ASCII consists of the 256 numbers assigned to the alphabet, numbers, punctuation, and other characters.

**bandwidth** Generally, the same as data transfer rate but also specifically the number of bits that can be sent through a network connection, measured in a second (bps); and the range of transmission frequencies a device can use, measured in Hertz (Hz) or cycle per second.

**bitmap** A graphic file that contains a record of the color value of each pixel in the graphic.

**bus** The circuitry and chips that manage the transfer of data from one device to another. A PC's motherboard has a bus, such PCI or PCI-Express, but there are also buses between memory and the processor and external components, such as a SCSI or universal serial bus.

**cable** A collection of wires separated by insulation that carry electrical signals between components. They can be inside or outside a PC.

**capacitance** A measure of the charge between two electrical plates that are separated by a non-conductive dialectic material. Because capacitance changes depending on the proximity, shape, and size of the plates, dialectic and even nearby object capacitance often is used to measure input.

**circuit, circuit board** Metallic traces printed on a fiberglass panel that carry electrical signals among larger components.

**contact** The metal pins and tabs where electrical circuits from two components make physical contact so electricity can flow from one to the other.

**charge-coupled devices (CCDs)** Miniature devices that convert the energy from light into electrical current. CCDs are used in digital cameras and scanners.

**CRT (cathode ray tube)** The "picture tube" of a traditional TV set or a computer monitor. Rays of electrons from a cathode in the back of the tube are guided along lines of phosphors on the inside front of the tube that glow when the electrons energize them.

**data transfer rate** The amount of data that moves from one device to another in a specific amount of time.

**digital-to-analog converter (DAC)** A chip that converts a series of digital values into a smoothly varying analog electrical current.

**dot pitch** The distance between the nearest two pixels of the same color. Smaller is better.

**expansion slot** A strip of connectors on a PC motherboard into which an adapter card is plugged.

**input/output** Input is data flowing into a device; output is data flowing out. Note that one device's output is another's input. You can't have one without another.

**ISA** Acronym for Industry Standard Architecture, referring to a 16-bit bus/expansion slot design that has been the most common type of adapter slot.

**LCD (active matrix)** A flat display screen using liquid crystals. It uses individual electrodes leading to each pixel, resulting in a bright image.

**PCI** Acronym for Peripheral Component Interconnect, a 32-bit successor to the ISA bus.

**pixel** Derived from "picture element"—the smallest unit of a computer display. A *physical pixel* equals the physical size of the dot pitch and consists of only one dot each of red, blue, and green. Several physical pixels can go into making up one *logical pixel*, the smallest grouping of color dots that a video card works with as if they were a single point of light.

**polarization** Light is ordinarily non-polarized. That means on a straight line leading away from a light source, light waves will be vibrating at all angles to the line. Polarization happens when light passes through a filter that permits the light waves to pass only if their plane of vibration is within a limited range.

**port** Generally, a specific place where one device, usually represented by a cable or wire, is physically connected to some other device through a socket or slot. In a broader sense, any point at which this data is transferred.

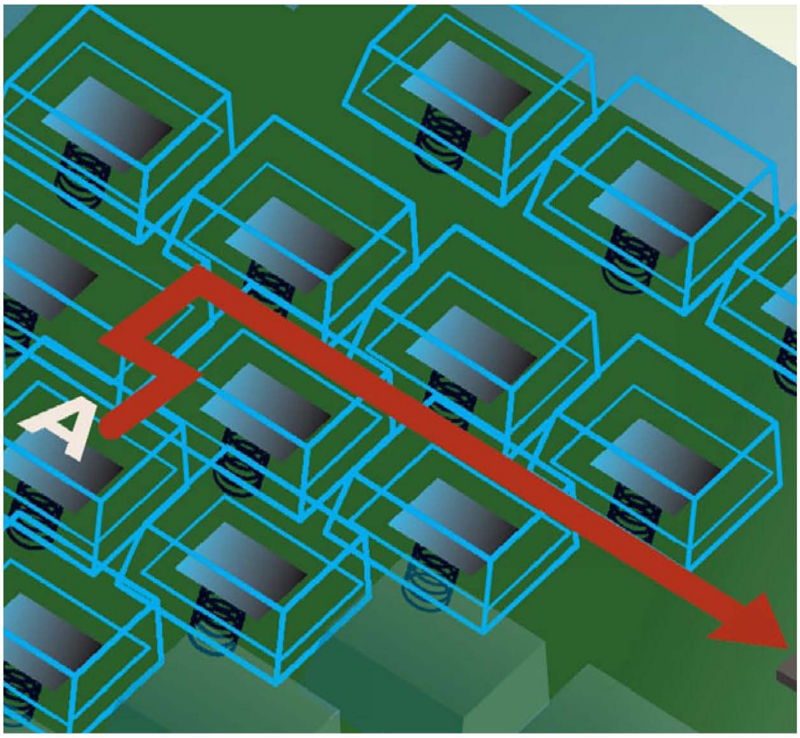
**serial/parallel** A serial port or connection is one in which only one bit of information can be sent at a time because only one wire or path is used for data. A parallel connection allows several bits—usually at least eight—to be transmitted simultaneously along separate wires.

**throughput** The amount of work a computer or component can do in a specific amount of time.

**VGA, super VGA** VGA is a specific standard of monitor resolution—640 pixels horizontally and 480 pixels vertically—and a limited range of colors. A super VGA monitor is capable of displaying resolutions of 800×600, 1028×782 pixels and higher. The number of colors in a super VGA display range from 256 to several million.

CHAPTER 14

# How Data Gets Into Your PC



**YOU** come into direct contact with your PC's keyboard more than you do any other component. You might go for years without ever thinking about—much less touching—your PC's processor or hard drive, but most people pay much more attention to those components than they do to the one part of the computer that determines not how well the computer works, but how well they themselves work.

A poorly designed keyboard acts as a constant stumbling block to productivity and can even cause health problems. A well-designed keyboard is one that you never think about; your thoughts seem to flow directly from your mind to the computer's screen without you being aware of what your fingers are doing.

Despite the importance of the keyboard, most manufacturers—and too many users—pay little attention to it. In 1996, Microsoft made the biggest change in the keyboard since the function keys moved from the left side to the top: Microsoft's split-board design made a concession to ergonomics by splitting the layout in half and angling the halves so they're in line with how our arms rest naturally on a desktop. The design has been widely copied, but it's unlikely to completely replace the older arrangement that most people are used to.

Regardless of changes in layout, the basic way a keyboard works has not changed significantly since the first IBM PC was introduced in the early 1980s.

Unfortunately, there is nothing natural or intuitive about a keyboard. No child is born knowing how to type, and even when the skill is learned, there's little sense to it—no one can give a sensible explanation of why the alphanumeric keys are arranged the way they are.

For many, the keyboard is actually a barrier to learning how to use a computer. Even for the experienced typist, there's nothing instinctive in pressing F5 to print a file. Engineers—not one of them touch typists, I'll bet—at Xerox Corporation's Palo Alto Research Center (PARC) developed a concept first explored by Douglas C. Engelbert of the Stanford Research Center. The concept was a **pointing device**, something a computer user could move by hand, causing a corresponding movement onscreen. Because of its size and tail-like cable, the device was named for the mouse. Apple Computer made the mouse a standard feature of its Macintosh computers, and Windows has made a mouse standard equipment on PCs, as well.

The mouse is not the only pointing device that's been invented. Digitizing tablets are popular with artists and engineers who must translate precise movements of a pen into lines on the screen. The most successful pointing innovations have been "eraserhead" pointing devices, so called because they look like a pencil's eraser stuck between the G and H key; the touch pad, which is a digitizing table without the precision; and trackballs. All three are popular on laptops, used where there's no space for a conventional mouse.

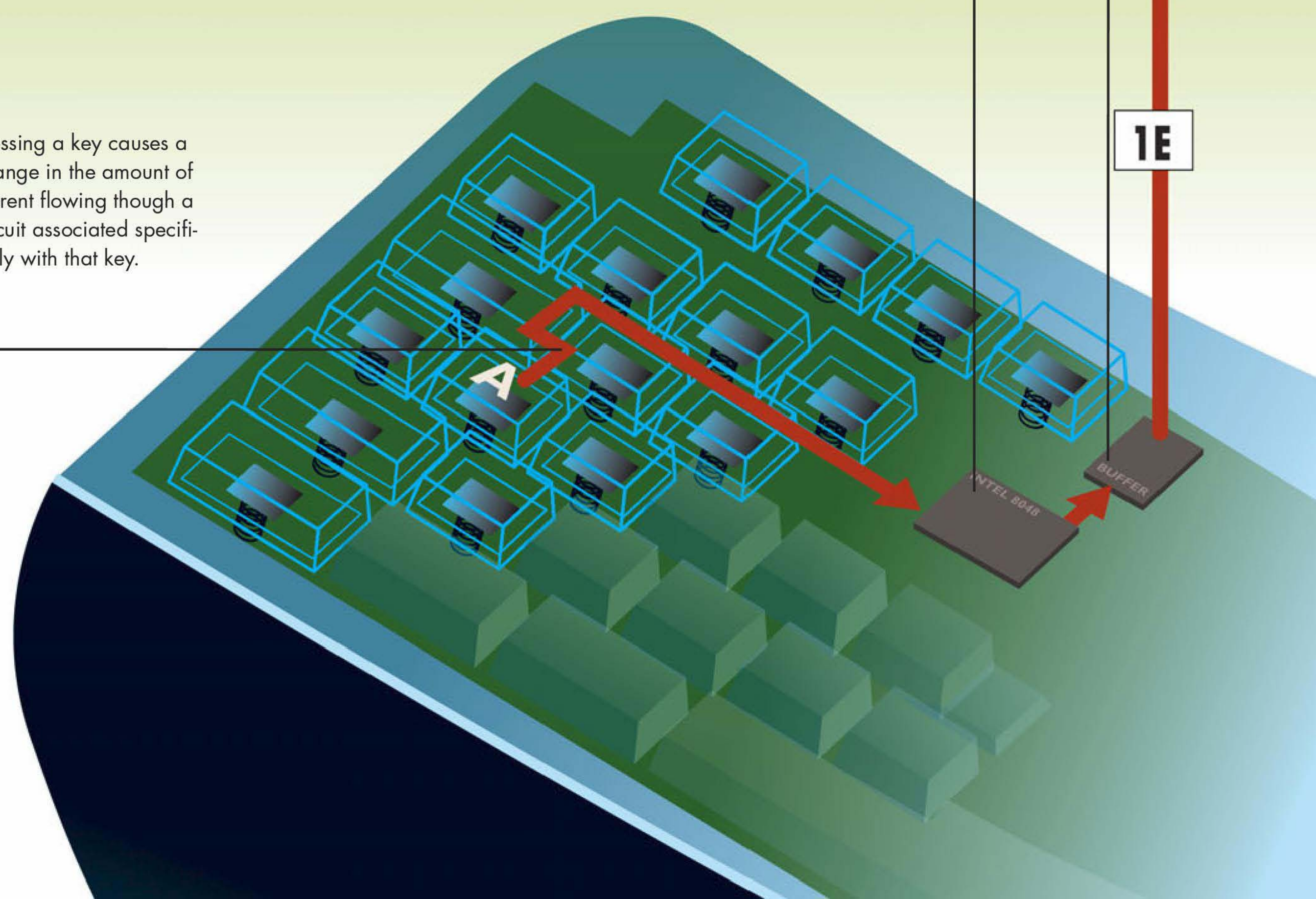
The mouse and its cousins can never replace the keyboard, but they can supplement the keys by doing tasks such as moving and pointing to onscreen objects, tasks for which the cursor keys are ill-suited. We're only just reaching the point where we control our PCs simply by speaking to them.

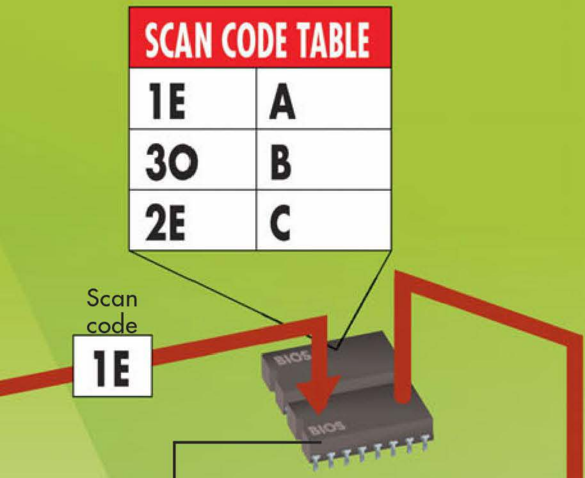
# The Keyboard and Scan Codes

**3** Depending on which key's circuit carries a signal to the microprocessor, the processor generates a number, called a **scan code**. There are two scan codes for each key, one for when the key is depressed and the other for when it's released. The processor stores the number in the keyboard's own memory buffer, and it loads the number in a port connection where it can be read by the computer's **BIOS** (basic input/output system). The processor then sends an interrupt signal over the keyboard cable to tell the processor that a scan code is waiting for it. An interrupt tells the processor to drop whatever else it is doing and to divert its attention to the service requested by the interrupt.

**2** A microprocessor built into the keyboard constantly scans circuits leading to the keys. It detects the increase and decrease in current from the key that has been pressed. By detecting both an increase and a decrease in current, the processor can tell when a key has been pressed and when it has been released. Each key has a unique set of codes, even if, to the users, the keys seem identical. The processor can, for example, distinguish between the left and right shift keys. To distinguish between a real signal and an aberrant current fluctuation, the scan is repeated hundreds of times each second. The processor only acts upon signals detected for two or more scans.

**1** Pressing a key causes a change in the amount of current flowing though a circuit associated specifically with that key.

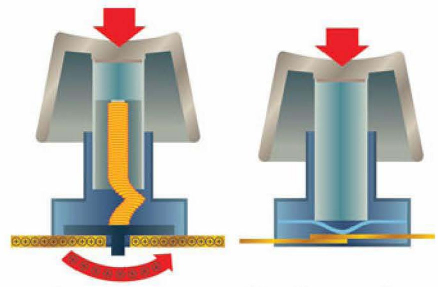


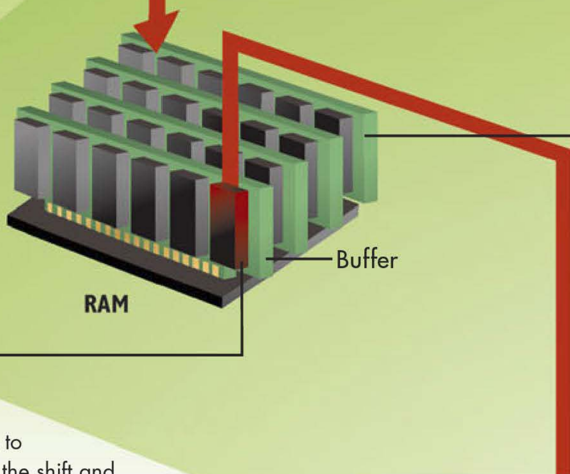



**4** The BIOS reads the scan code from the keyboard port and sends a signal to the keyboard that tells the keyboard it can delete the scan code from its buffer.

## Keys to the PC

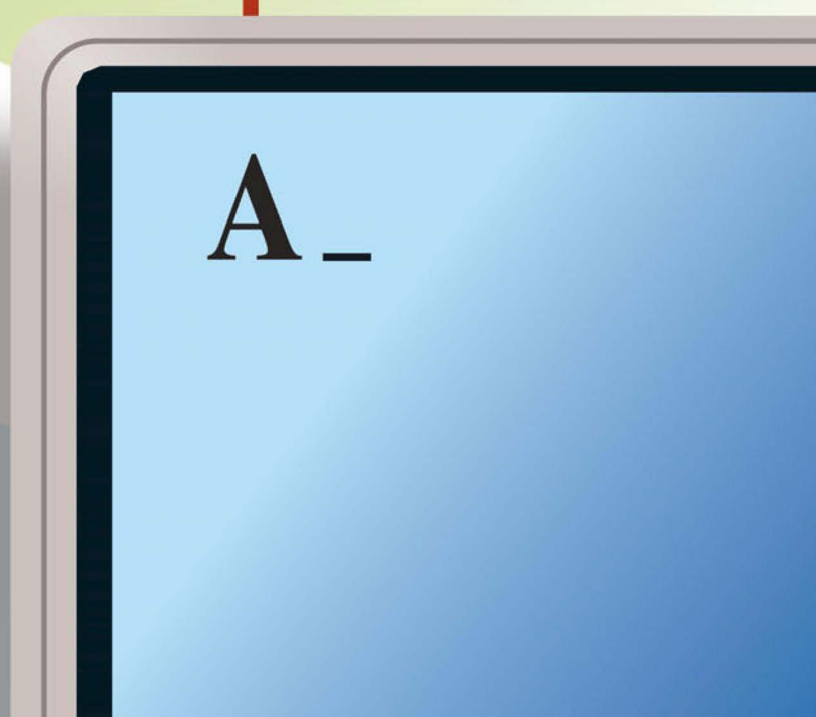
Two types of keys are used on keyboards. **Capacitive** keys are built around a spring that makes a clicking noise when the key is depressed. Pressing it causes a metal plunger to pass between two metal pads on the underlying circuit boards that act as a capacitor. The plunger causes a change in the electrical potential between the two pads, which signals that the key is pressed down. **Hard-contact** keys are mounted above a rubber dome. Pressing the key collapses the dome and presses two metal plates together so current flows through them. When the key is released, the dome pops the keycap back up.




**5** If the scan code is for one of the ordinary shift keys or for one of the special shift keys and toggle keys—Ctrl, Alt, Num Lock, Caps Lock, Scroll Lock, or Insert—the BIOS changes two bytes in a special area of memory to maintain a record of which of these keys has been pressed.

**6** For all other keys, the BIOS checks those two bytes to determine the status of the shift and toggle keys. Depending on the status indicated by those bytes, the BIOS translates the appropriate scan code into an ASCII code, used by the PC, that stands for a character, or into a special code for a function key or a cursor movement key. Uppercase and lowercase characters have different ASCII codes. Applications can choose to interpret any keystroke to display a character, or as a command. Ctrl+B, for example, is universally used by Windows applications to toggle the boldface attribute. In either case, the BIOS places the ASCII or special key code into its own memory buffer, where it is retrieved by the operating system or application software as soon as any current operation is finished.

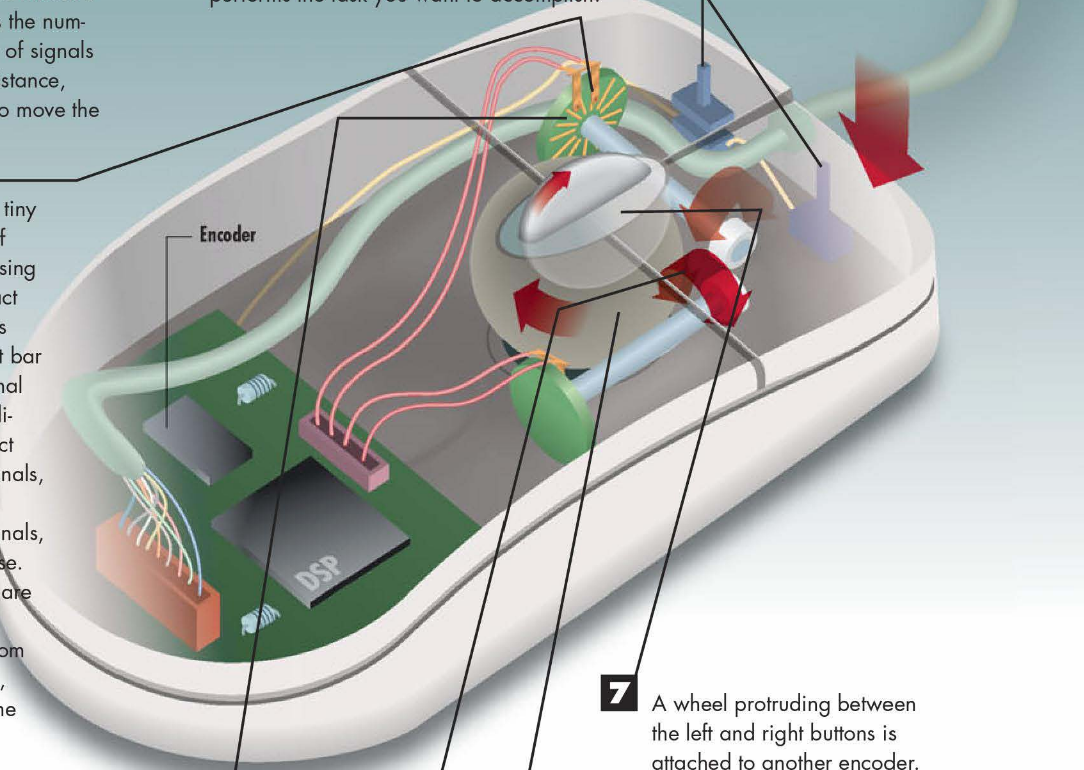


# How Mice Obey Your Every Gesture

## Mechanical Mouse

**5** Signals are sent to the PC over the mouse's tail-like cable. Windows converts the number, combination, and frequency of signals from the two encoders into the distance, direction, and speed necessary to move the onscreen cursor.

**6** Tapping either of the buttons atop the mouse also sends a signal to the PC, which passes the signal to the software. Based on how many times you click, and the position of the onscreen pointer at the time of the click, the software performs the task you want to accomplish.




**4** On the rims of each encoder are tiny metal contact points. Two pairs of contact bars extend from the housing of the mouse and touch the contact points on each of the encoders as they pass by. Each time a contact bar touches a point, an electrical signal results. The number of signals indicates how many points the contact bars have touched—the more signals, the farther you have moved the mouse. The more frequent the signals, the faster you're moving the mouse. The direction in which the rollers are turning, combined with the ratio between the number of signals from the vertical and horizontal rollers, indicates the direction in which the mouse is moving.

**3** Each roller is attached to a wheel, known as an **encoder**, much as a car's drive train is attached by its axles to the wheels. As the rollers turn, they rotate the encoders.

**2** As the ball rotates, it touches and turns two rollers mounted at a 90° angle to each other. One roller responds to back-and-forth movements of the mouse, which correspond to vertical movements onscreen. The other roller senses sideway movements, which correspond to side-to-side movements onscreen.

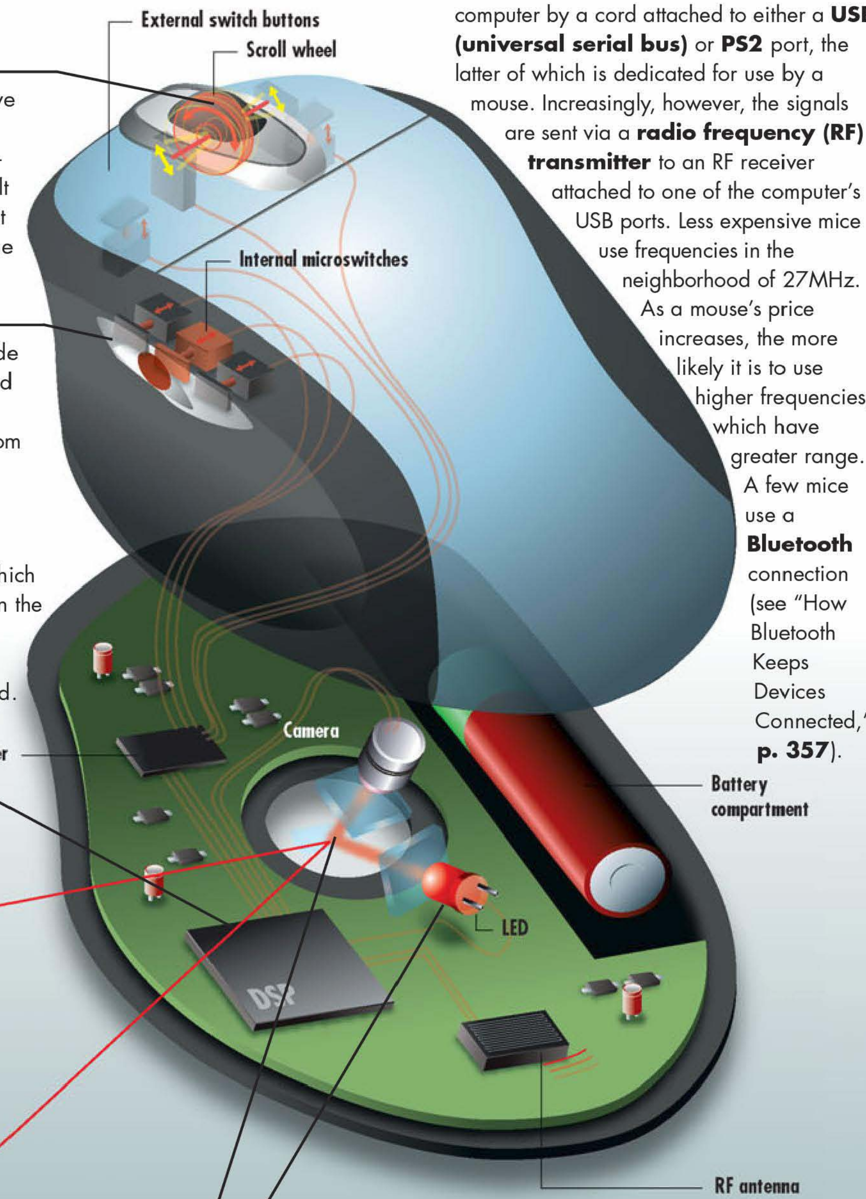
**1** As you move a mechanical mouse by dragging it across a flat surface, a ball—made of rubber or rubber over steel—protruding from the underside of the mouse turns in the direction of the movement.

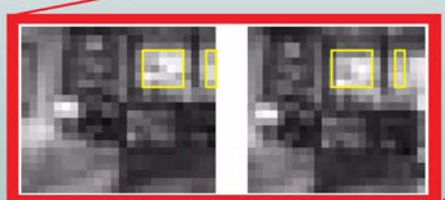
**7** A wheel protruding between the left and right buttons is attached to another encoder. As you move the wheel forward or backward, the encoder sends signals that the software interprets to scroll the screen up or down.

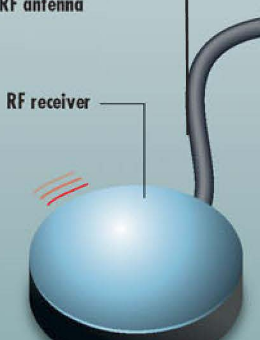
### A Mouse on Its Back

Want to know how a trackball works? Turn a mechanical mouse upside down and you'll get some idea. A trackball is simply a mouse mounted so the ball is rotated with your fingers instead of on the surface of your desk.

# Optical Mouse






**1** As you move an optical mouse, an **LED—light emitting diode—** lights up the surface the mouse is moving across on low-friction pads. An alternative to the LED is a laser. The basic operation is the same for either light source, although laser proponents claim greater resolution and responsiveness as well as the ability to work on polished surfaces that an LED mouse would find confusing.

**2** A digital camera about the size of a dime peers through a plastic lens at the surface lit up by the LED. The camera takes hundreds of photos a second, looking for differences among the images that indicate the speed and direction of the mouse.

**3** The camera sees the surface in only black and white. Here the surface's microscopic pattern has moved up and to the right from one frame to another, indicating that the mouse moved to the left and down.

**4** The signals from the camera are fed to a microprocessor called a **digital signal processor (DSP)** for interpretation. The DSP chews through as many as 4.7 megapixels a second.

**5** The information the DSP obtains is fed to the computer by a cord attached to either a **USB (universal serial bus)** or **PS2** port, the latter of which is dedicated for use by a mouse. Increasingly, however, the signals are sent via a **radio frequency (RF) transmitter** to an RF receiver attached to one of the computer's USB ports. Less expensive mice use frequencies in the neighborhood of 27MHz. As a mouse's price increases, the more likely it is to use higher frequencies, which have greater range. A few mice use a **Bluetooth** connection (see "How Bluetooth Keeps Devices Connected," **p. 357**).

**6** Whatever the transmission method, the information makes its way to the PC, which uses the data to reposition the mouse on the screen. This entire process results in the mouse sending reports on its current speed and direction 125 times a second.

**7** Several other buttons on the top and side of the mouse allow you to move forward or backward with an Internet browser, control the volume of music, or jump from one application to another.

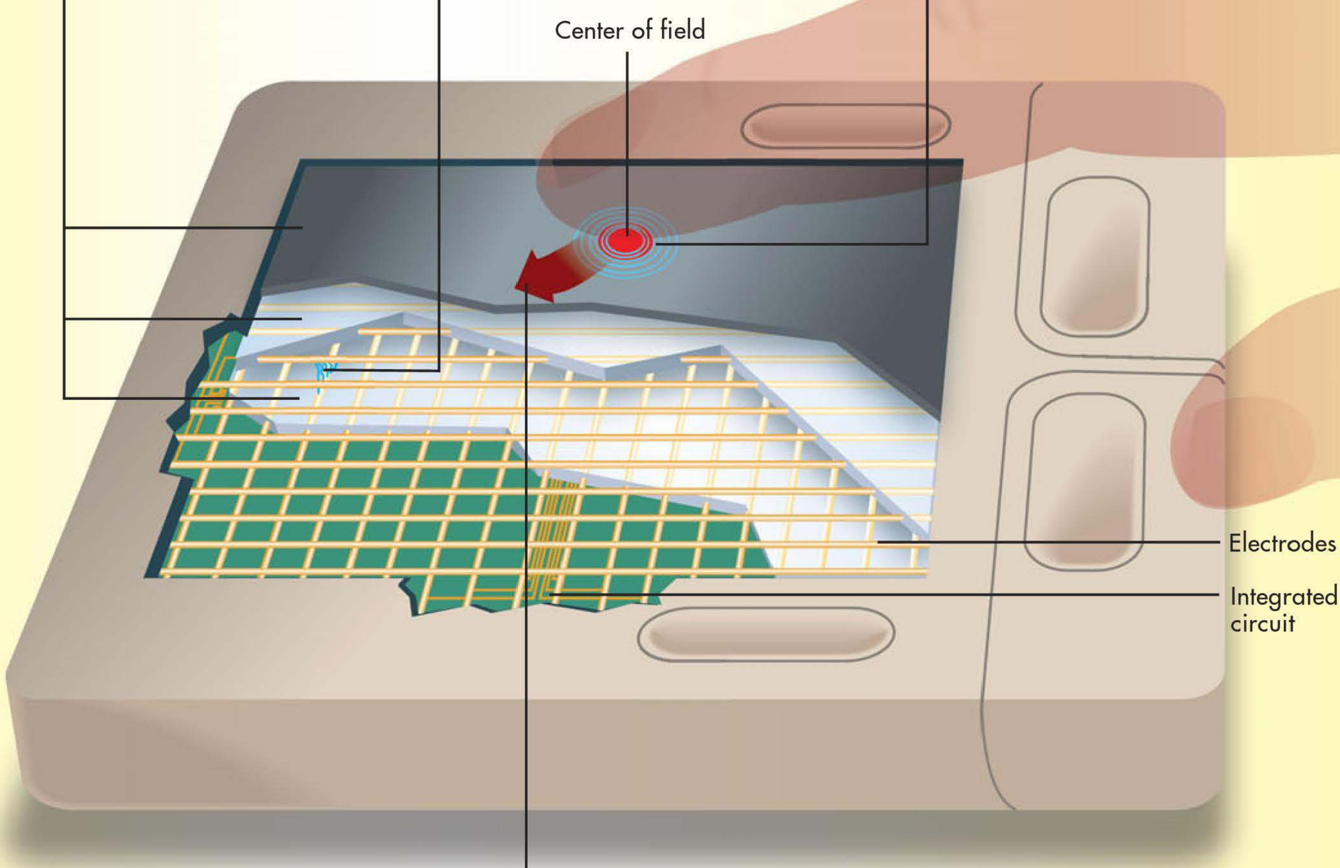
**8** Optical mice are usually more expensive than a mechanical mouse, and subsequently come with more bells and whistles. Its **scroll wheel** enables you to tilt it to the right or left so it presses against **microswitches** that send signals to the software telling it to scroll sideways.

# How a Touchpad Works

**1** Beneath the top rubber layer of a touchpad are two more layers, each of which contains a row of electrodes, one row going horizontally and the other vertically.

**2** The crossing electrodes do not touch, but a positive electrical charge builds up in one set and a negative charge in the other. This creates an electric field between the layers. Integrated circuits for the horizontal and vertical electrodes sample the strength of that field's electrical potential, or **mutual capacitance**. The size and shape of the electrodes and the nonconductive, dialectic material separating them influence the amount of capacitance.

**3** Capacitance is also affected by the surrounding electromagnetic field from other objects, including a finger, which has very different dialectic properties from air. Even if the finger doesn't actually touch the pad, the fingertip's field penetrates the grid of electrodes, changing the capacitances where electrodes cross over and under one another nearest the fingertip.

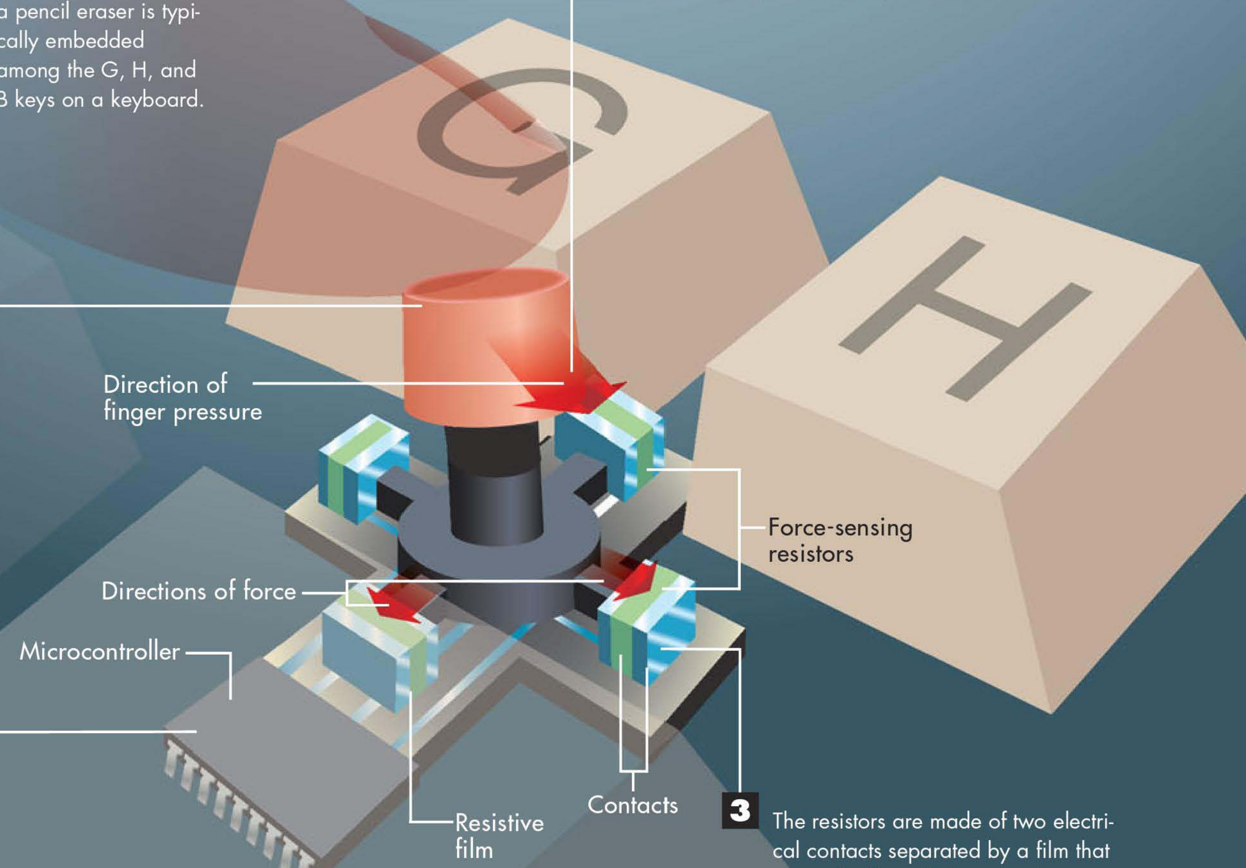



**4** The capacitances are most affected at the center of the finger. By reading the capacitances of adjoining intersections, the touchpad can identify the finger's center, and it feeds that location to Windows to position the onscreen arrow. The capacitances are measured about 100 times a second. Changes in those measurements caused by moving the finger are translated into cursor movement.

# How a Pointing Stick Works

**1** The portion of a pointing stick that looks like a pencil eraser is typically embedded among the G, H, and B keys on a keyboard.

**2** When a finger applies lateral pressure to the eraser head, it does not move. Instead, the force is passed on to a combination of four force-sensing resistors placed to measure forward, backward, and sideways forces.



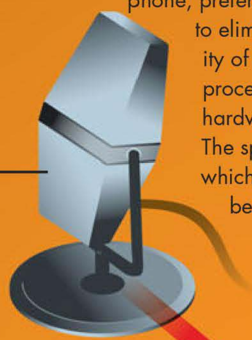

**3** The resistors are made of two electrical contacts separated by a film that resists the flow of electricity. Pressure from the finger is passed to one of the contacts, squeezing it against the film and creating a better connection between the contacts so that more electricity flows between the contacts.

**4** A microcontroller monitors the amounts of electricity passing through all the resistors and uses that information to translate the finger pressure into onscreen cursor movement. There are minimums and maximums to how much the currents can vary, which prevents runaway pointer movements caused by casually touching the pointing stick or by pressing it too hard.

# How Speech Recognition Works

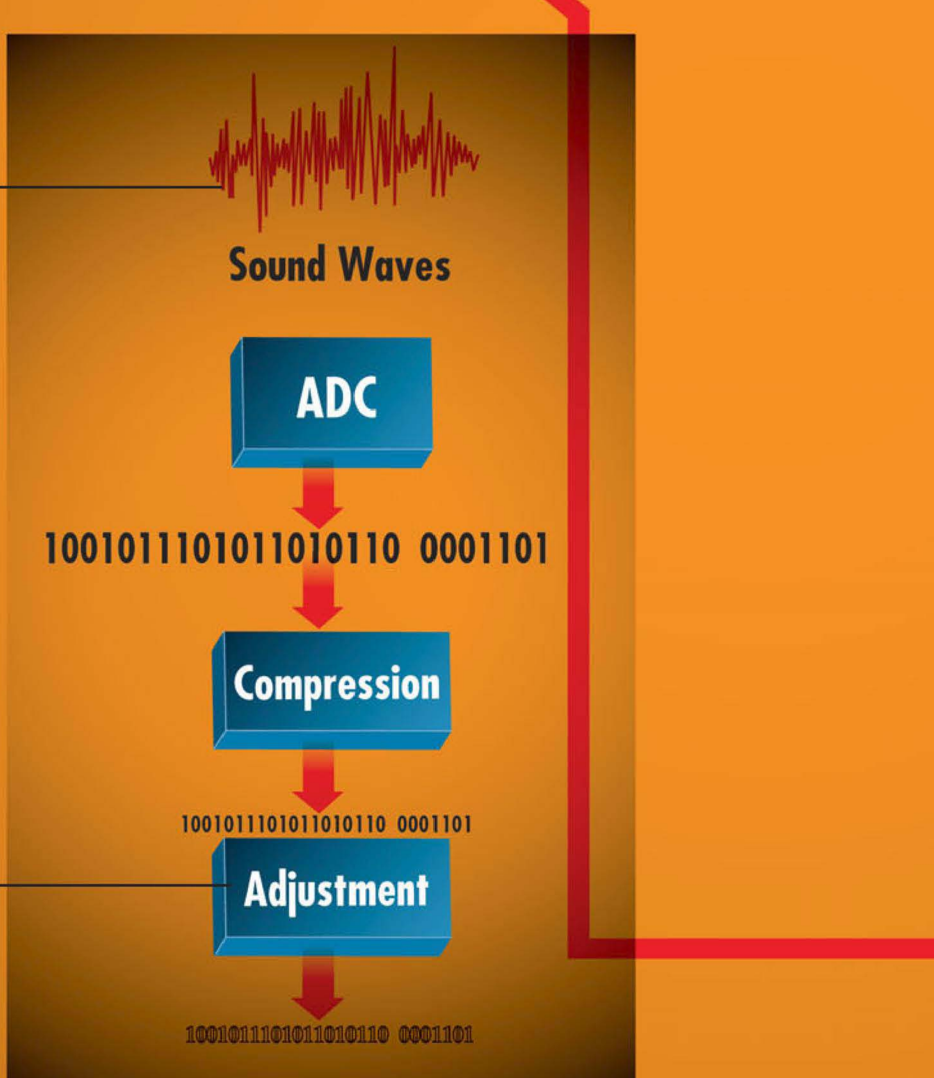
**1** A person who's going to use speech recognition software must first go through an **enrollment**. This consists of the person dictating text that is already known to the software for 10 minutes to an hour. From this sampling, the software creates a table of vocal references, which are the ways in which the speaker's pronunciation of phonemes varies from models of speech based on a sampling of hundreds to thousands of people. **Phonemes** are the smallest sound units that combine into words, such as "duh," "aw," and "guh" in "dog." There are 48 phonemes in English.

**2** After enrollment, the speaker dictates the text he wants the software to transcribe into a microphone, preferably one that uses noise-cancellation to eliminate background sounds. The quality of the microphone and the computer's processing power are the most important hardware factors in speech recognition. The speaker can use continuous speech, which is normal speech without pauses between words.



**3** Once every 10–20 milliseconds, the analog signal generated by the microphone is sampled by an analog-to-digital converter (ADC) that converts the spoken sound to a set of measurements of several factors, including pitch, volume, frequency, length of phonemes, and silences. The measurements are compressed for quicker processing.

**4** The software's **speech engine** makes adjustments to the phoneme measurements by factoring in background sounds, the acoustic characteristics of the microphone, and, from the table of vocal references, the speaker's individual idiosyncrasies of accent, regional pronunciations, and voice characteristics.



**Acoustic Recognizer**

| Models | Matches: Volume | Pitch | Length | Tremor |
|---|---|---|---|---|
| i: (as in six) | ✓ | | | |
| I (as in sit) | | ✓ | | |
| əʊ (as in ten) | ✓ | | ✓ | |
| æ (as in hat) | ✓ | ✓ | | ✓ |
| au (as in home) | | | | |
| ŋ (as in sing) | | ✓ | | |

**5** The **acoustic recognizer** compares the measurements of the corrected phonemes to a binary-tree database of compressed, known phoneme measurements—**models**—compiled from sampling the speech of thousands of people. For each measurement, such as pitch, the recognizer finds the database entries that most closely match that specific measurement. To narrow the selection further, the engine compares another measurement—for example, volume—to the volume measurement of only those database models that received a high score on the pitch measurement. The process continues until the engine finds the model phoneme that most closely matches the sample phoneme across the entire range of measurements.

**Lexicon**

| | Matches |
|---|---|
| th e r | there, their, they're |

**6** To make words out of the phonemes, the speech engine compares groups of successive phonemes to a database of known words. Typically, tens of thousands of words make up the lexicon. Speech recognition for specialized applications, such as medical or legal dictation, uses a database customized with words unique to that subject.

**Natural Language Engine**

... Put it their.

... Put it there. ✓

... Put it they're.


**7** In the case of homonyms—words that sound alike but are spelled differently, such as "there" and "their," the results are turned over to a natural language component, which compares the sounds with grammatical rules, a database of most common phrases, and other words used in the context of the spoken text. The natural language rules predict the most likely word combination to occur at each point in a sentence, based on the relative frequency of all groups of three words in the language. For example, it knows "going to go" is more frequent than "going, too, go" and "going two go." It displays onscreen the word combination that has the highest probability of matching what the speaker said.

**8** If the speech engine cannot resolve some ambiguity, it might query the speaker, presenting a list of candidate words and allowing the speaker to choose the correct one. Or, if the engine chooses the wrong word, the speaker corrects the choice, and the change is fed back into the vocal references table of idiosyncrasies to improve the accuracy of future speech sessions.

CHAPTER

15

# How Scanners Capture Words and Images

**THE** problem with computers from the very start was that before they could do all their marvelous tricks with math and information, you had to input some data with which they could work. Whole warehouses full of people were hired to sit at a keyboard for eight hours a day to input—the new word for typing—information that, for the most part, they were reading off forms onto which other people had already input information about themselves.

What's wrong with this picture? How could it be that we were capable of creating machines that could do in minutes (seconds!) what would take a university full of mathematicians 1,000 years to complete—and that's without spring breaks. It was swiftly becoming clear that the computers got all the fun assignments, like calculating the trajectory of artillery shells, while we humans were turning into drudges typing census information on punch cards that would then be fed to the computer so it could do the really important work.

The obvious solution to such indignities was to make the computers do their own input. See how *they* like typing—**inputting**—thousands of characters a day as if they were…machines! Of course, the computers always had the perfect comeback: "Sorry. No eyes." Can't argue with that—unless you had a tiny device that could measure light and, say, convert those measurements into electrical currents that could be fed into a computer somehow so it could "see" as well as humans. That tiny device, the photodiode, had actually been around since the 1970s. It just took a couple of decades for the miracle of seeing machines to really get geared up.

Part of the delay was due to the lack of software to take advantage of this new vision. Optical character recognition is no easy trick. It takes humans a few years to get the hang of it. But scanners are now capable of translating printed text in virtually any shape, form, or layout into editable text. The search engine company Google is currently on a project to convert—to the extent that they are allowed to—every public domain book and paper in major libraries into editable, computer-searchable text. Scholarship will never be the same. Neither will war, now that computer eyes search for our enemies and guide our rockets.

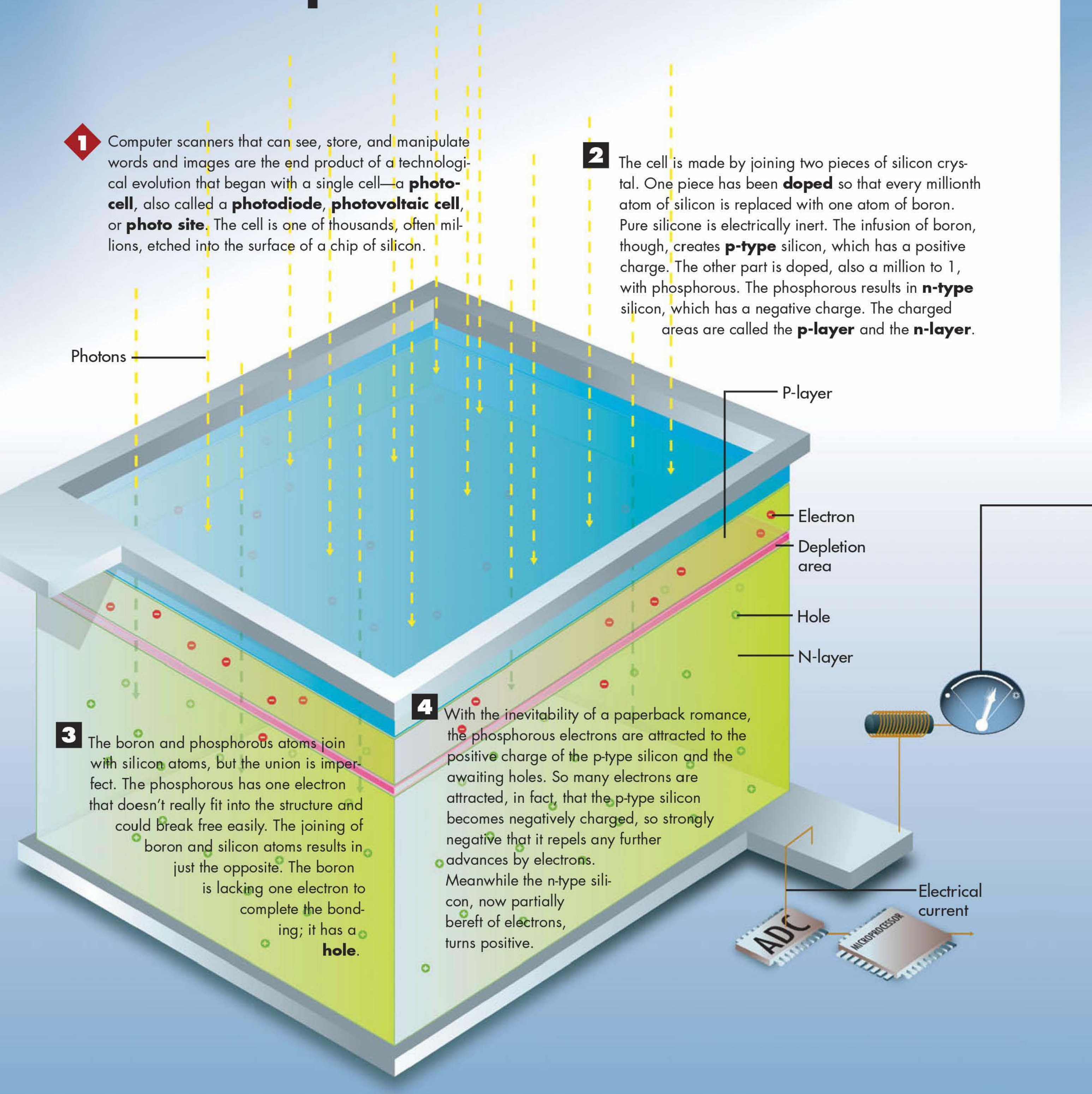
As scanning software got better and the PCs more powerful and able to see sharper images, scanners began appearing in businesses and in homes, where a scanner could become an ad-hoc Xerox as well as save priceless family photos that were slowly decomposing.

Now, as an integral part of digital photography and digital darkrooms, the computer's eyes have let it become our artistic vision—and our forger, as software allows the most clumsy draftsman to do photo touch-ups that are nearly undetectable. Who would have thought computers would become so good at being our eyes?

# How Computers See

**1** Computer scanners that can see, store, and manipulate words and images are the end product of a technological evolution that began with a single cell—a **photocell**, also called a **photodiode**, **photovoltaic cell**, or **photo site**. The cell is one of thousands, often millions, etched into the surface of a chip of silicon.

**2** The cell is made by joining two pieces of silicon crystal. One piece has been **doped** so that every millionth atom of silicon is replaced with one atom of boron. Pure silicone is electrically inert. The infusion of boron, though, creates **p-type** silicon, which has a positive charge. The other part is doped, also a million to 1, with phosphorous. The phosphorous results in **n-type** silicon, which has a negative charge. The charged areas are called the **p-layer** and the **n-layer**.

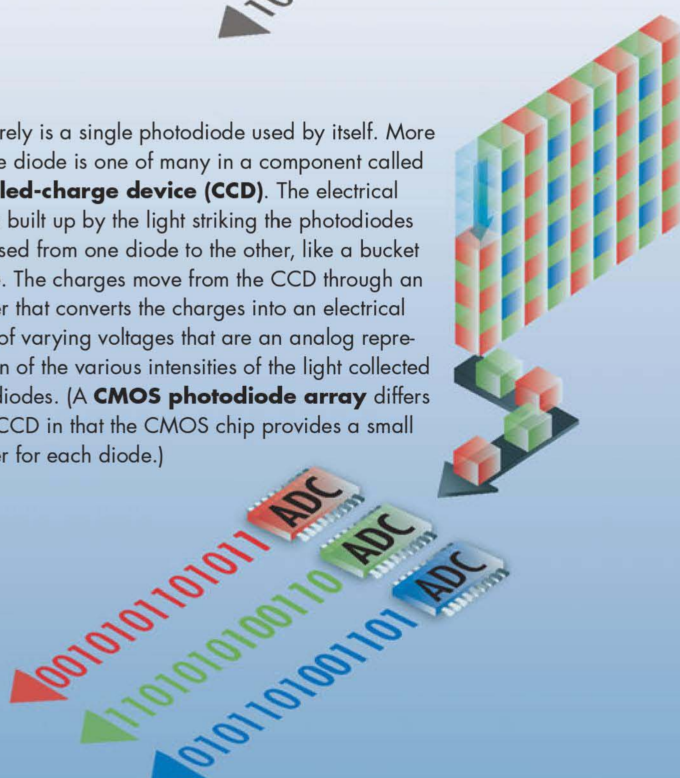



**3** The boron and phosphorous atoms join with silicon atoms, but the union is imperfect. The phosphorous has one electron that doesn't really fit into the structure and could break free easily. The joining of boron and silicon atoms results in just the opposite. The boron is lacking one electron to complete the bonding; it has a **hole**.

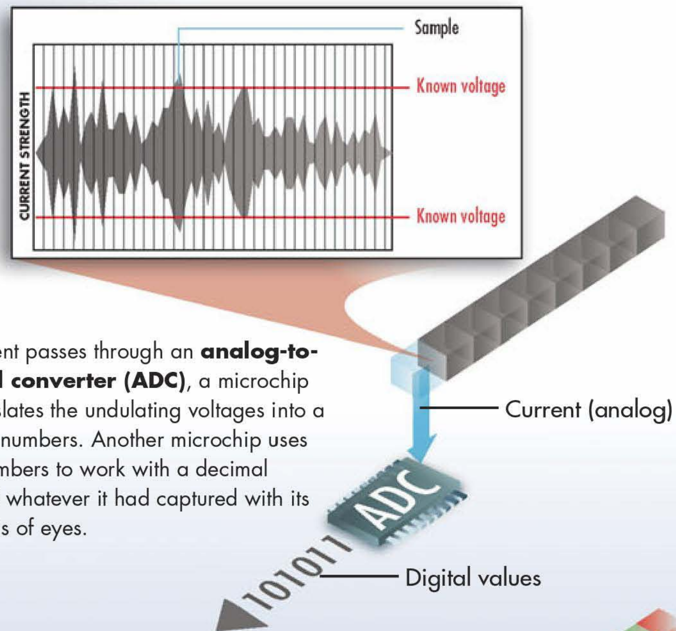
**4** With the inevitability of a paperback romance, the phosphorous electrons are attracted to the positive charge of the p-type silicon and the awaiting holes. So many electrons are attracted, in fact, that the p-type silicon becomes negatively charged, so strongly negative that it repels any further advances by electrons. Meanwhile the n-type silicon, now partially bereft of electrons, turns positive.

**5** This static stand-off ends when **photons**, the particle form of light, strike a photodiode. The light's energy is transferred to electrons, creating a negative charge. The negative charge and the positive charge of the n-layer create a magnetic field in which electrons from the p-layer are drawn by the positive force into the diode's **depletion area**, a narrow strip separating the negative and positive layers. The stronger the light that hits a pixel, the more electrons travel to the depletion layer, creating an electrical charge whose strength is proportional to the strength of the light that struck that particular photodiode.

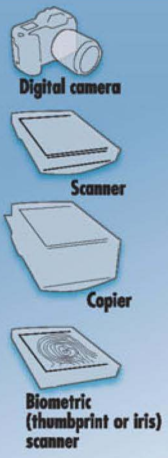
**6** Only rarely is a single photodiode used by itself. More often the diode is one of many in a component called a **coupled-charge device (CCD)**. The electrical charges built up by the light striking the photodiodes are passed from one diode to the other, like a bucket brigade. The charges move from the CCD through an amplifier that converts the charges into an electrical current of varying voltages that are an analog representation of the various intensities of the light collected by the diodes. (A **CMOS photodiode array** differs from a CCD in that the CMOS chip provides a small amplifier for each diode.)



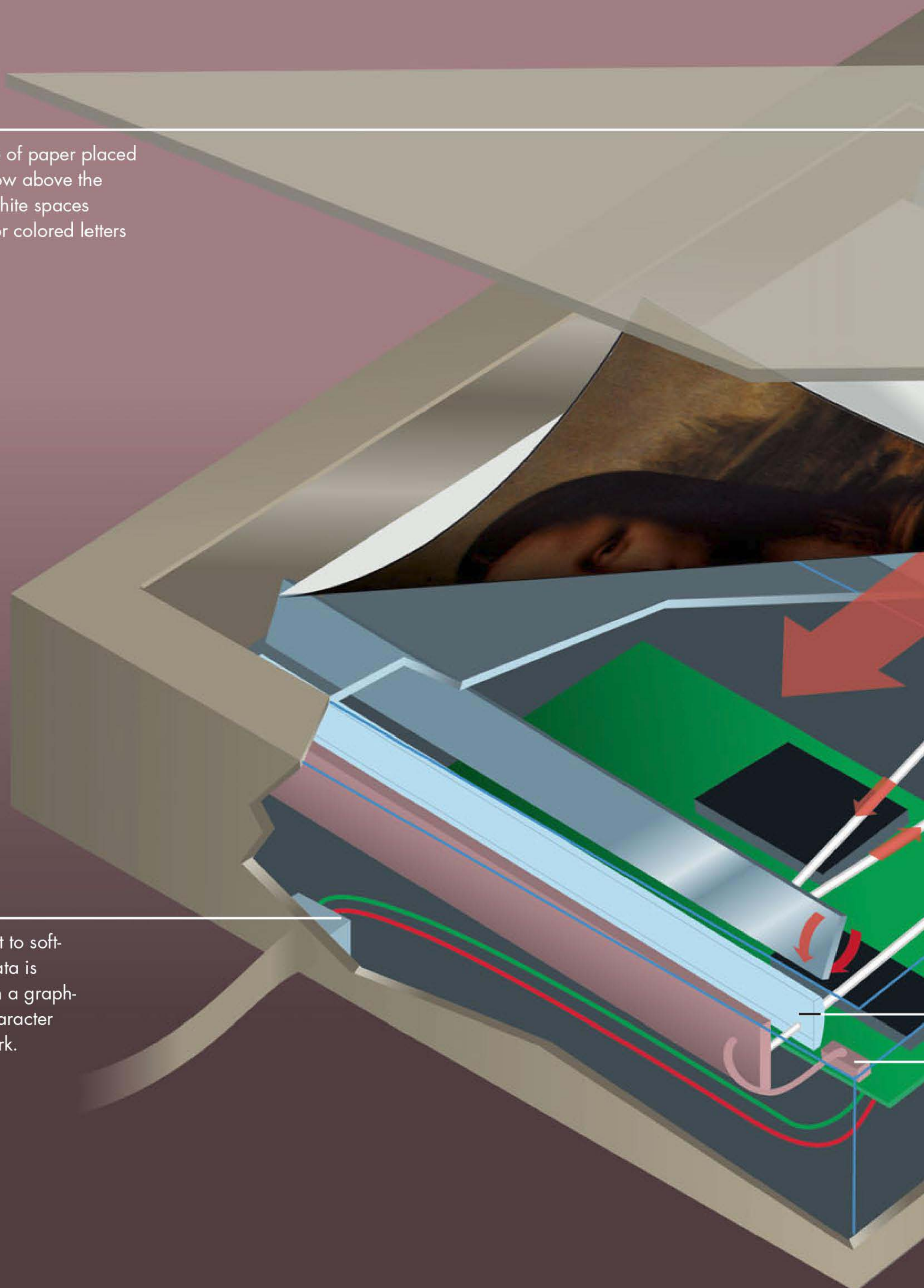

**7** The current passes through an **analog-to-decimal converter (ADC)**, a microchip that translates the undulating voltages into a series of numbers. Another microchip uses those numbers to work with a decimal image of whatever it had captured with its thousands of eyes.




**8** Light-detecting chips built on the CCD and CMOS model turn up in a variety of computer peripherals, from digital cameras to scanners, copiers, and fax machines, to biometric identification devices that can visually identify people through their fingerprints, retina scans, and facial characteristics.

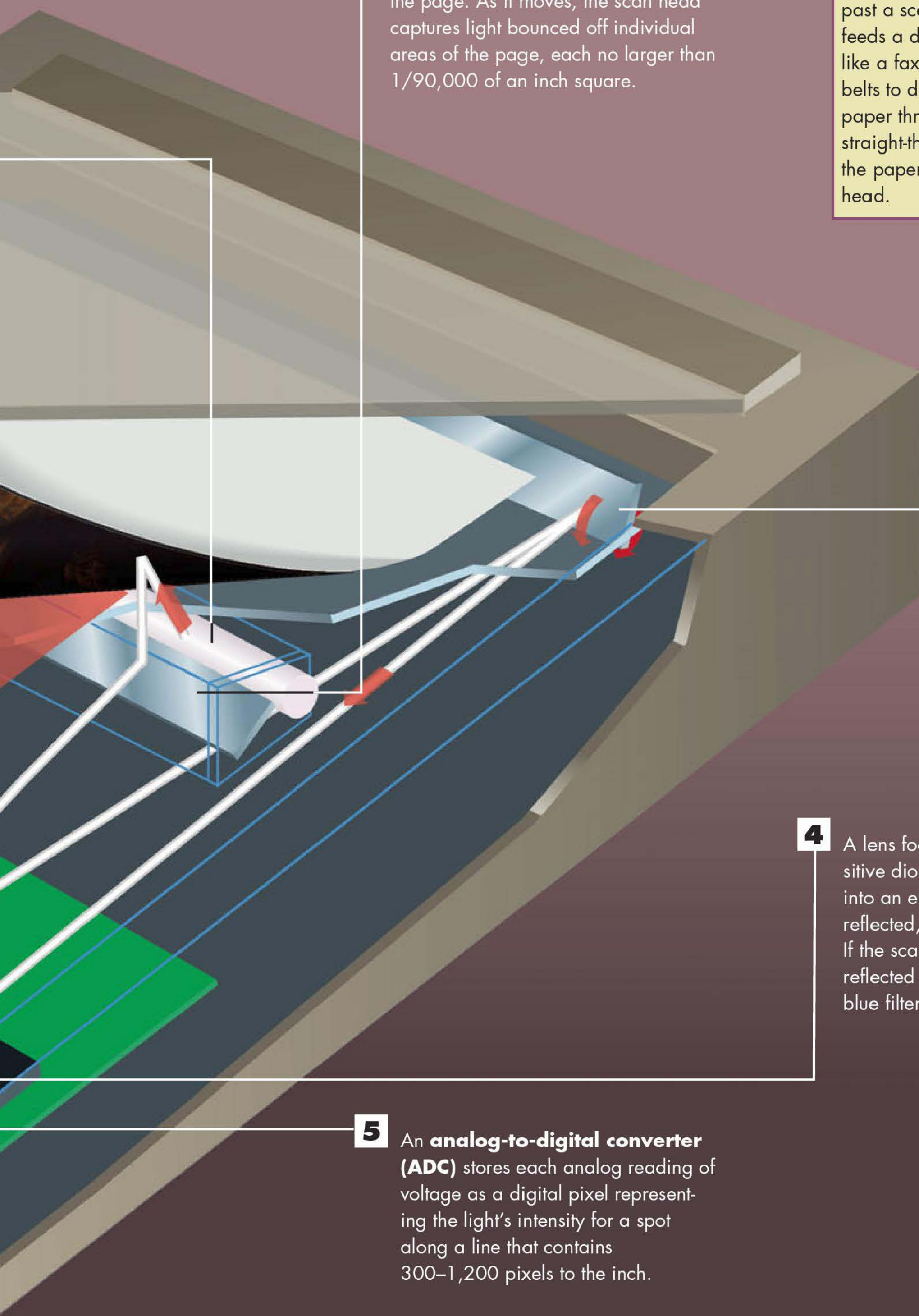


# How a Flatbed Scanner Works



**1** A light source illuminates a piece of paper placed face down against a glass window above the scanning mechanism. Blank or white spaces reflect more light than do inked or colored letters or images.

**6** The digital information is sent to software in the PC, where the data is stored in a format with which a graphics program or an optical character recognition program can work.



**2** A motor moves the scan head beneath the page. As it moves, the scan head captures light bounced off individual areas of the page, each no larger than 1/90,000 of an inch square.

### Paper Movers

The paper isn't always stationary when it's being scanned. Higher-end scanners feed sheets of paper past a scan head that stays still. A roller transport feeds a document between two rubber rollers, much like a fax machine. A belt transport uses two opposing belts to do the same thing. Drum transports pass the paper through against a rotating drum. A vacuum straight-through transport uses vacuum tubes to hold the paper against a belt as the belt rolls past the scan head.

**3** The light from the page is reflected through a system of mirrors that must continually pivot to keep the light beams aligned with a lens.

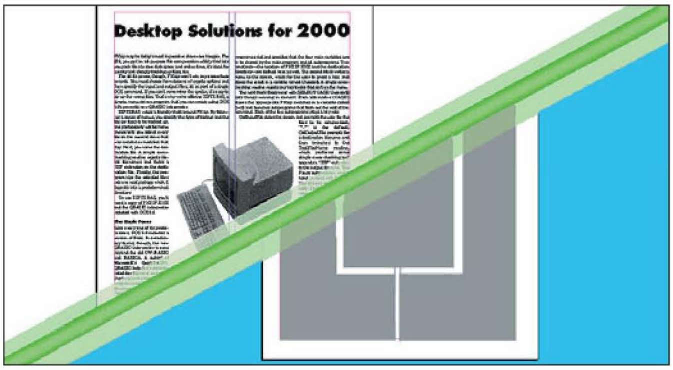
**4** A lens focuses the beams of light onto light-sensitive diodes that translate the amount of light into an electrical current. The more light that's reflected, the greater the voltage of the current. If the scanner works with colored images, the reflected light is directed through red, green, or blue filters in front of separate diodes.

**5** An **analog-to-digital converter (ADC)** stores each analog reading of voltage as a digital pixel representing the light's intensity for a spot along a line that contains 300–1,200 pixels to the inch.

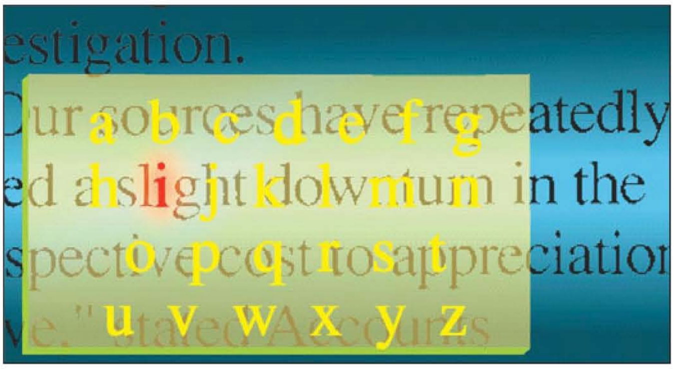
# How Optical Character Recognition Works

**1** When a scanner reads the image of a document, the scanner converts the dark elements—text and graphics—on the page to a **bitmap**, which is a matrix of square pixels that are either on (black) or off (white). Because the pixels are larger than the details of most text, this process degenerates the sharp edges of characters, much as a fax machine blurs the sharpness of characters. This degradation creates most of the problems for optical character recognition (OCR) systems.

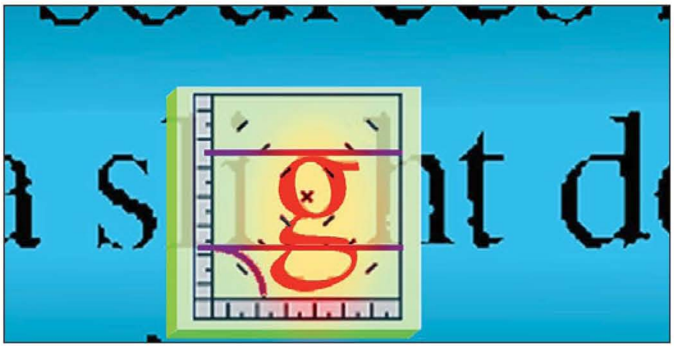

**2** The OCR software reads the bitmap that the scanner created and averages out the zones of on and off pixels on the page, in effect mapping the whitespace on the page. This enables the software to block off paragraphs, columns, headlines, and random graphics. The whitespace between lines of text within a block defines each line's baseline, an essential detail for recognizing the characters in the text.




**3** In its first pass at converting images to text, the software tries to match each character through a pixel-by-pixel comparison to character templates that the program holds in memory. Templates include complete fonts—numbers, punctuation, and extended characters—of such common faces as 12-point Courier and the IBM Selectric typewriter set. Because this technique demands a very close match, the character attributes, such as bold and italic, must be identical to qualify as a match. Poor-quality scans can easily trip up matrix matching.

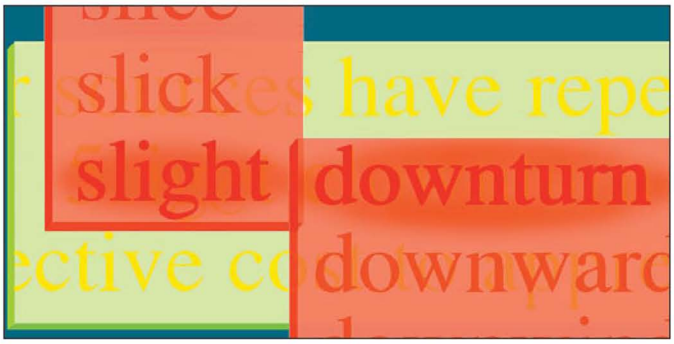


**4** The characters that remain unrecognized go through a more intensive and time-consuming process called **feature extraction**. The software calculates the text's **x-height**—the height of a font's lowercase x—and analyzes each character's combination of straight lines, curves, and **bowls** (hollow areas within loops, as in o or b). The OCR programs know, for example, that a character with a curved descender below the baseline and a bowl above it is most likely a lowercase g. As the software builds a working alphabet of each new character it encounters, recognition speed accelerates.

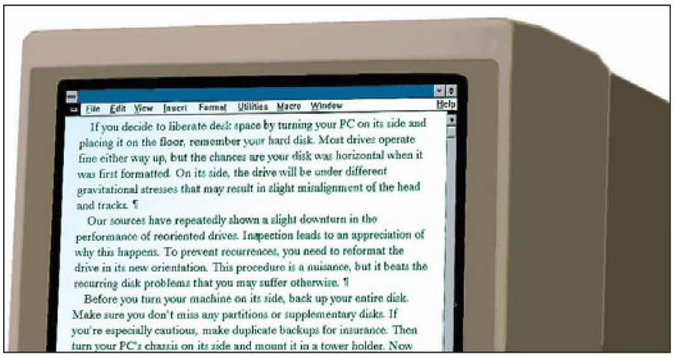


**5** Because these two processes don't decipher every character, OCR programs take two approaches to the remaining hieroglyphics. Some OCR programs tag unrecognized characters with a distinctive character—such as ~, #, or @—and quit. You must use the search capability of a word processor to find where the distinctive character has been inserted and correct the word manually. Some OCR programs also display a magnified bitmap onscreen and ask you to press the key of the character needed to substitute for the placeholder character.

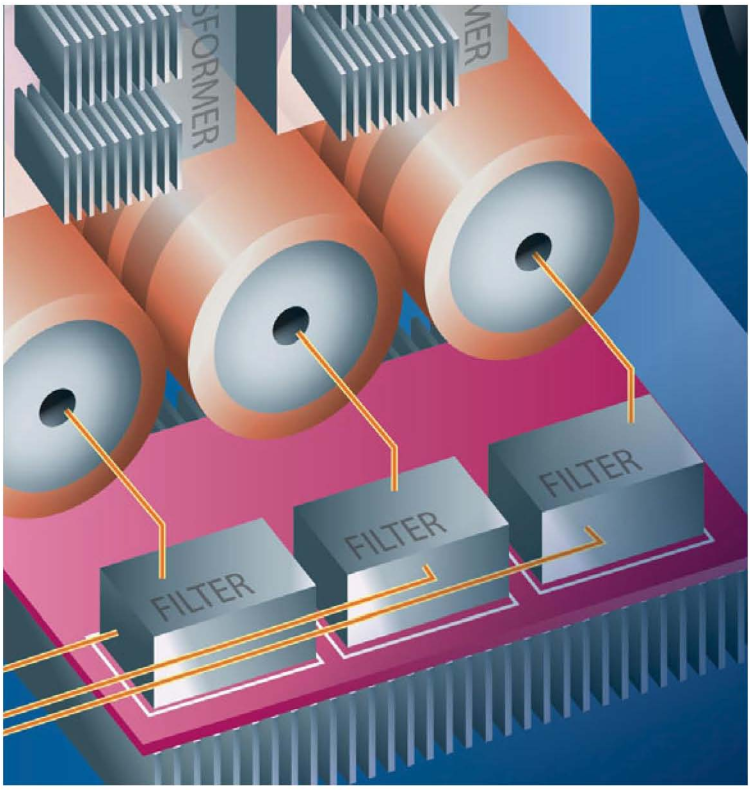

**6** Still other OCR programs invoke a specialized spelling checker to search for obvious errors and locate possible alternatives for words that contain tagged unrecognized characters. For example, to OCR programs, the number 1 and the letter l look very similar, so do 5 and S, or cl and d. A word such as *downturn* might be rendered as *clownturn*. A spelling checker recognizes some typical OCR errors and corrects them.




**7** Most OCR programs give you the option of saving the converted document to an ASCII file or in a file format that popular word processors or spreadsheets can recognize.



# CHAPTER 16 How Computers Use Power



**THINGS** are not as they seem. We're pretty smug in our universe, where events unfold evenly and seamlessly. Cars don't simply stop; they come to a stop. It takes time. A pendulum swings smoothly and predictably. Any time something moves from A to C, it passes an infinity of points B along the line. The light at the end of the tunnel gradually grows brighter. We are so comfortable with a perception of the world that is smooth and continuous, we've learned not to notice the finite increments of space and time in which things really do happen. We know that moving pictures are not moving; they appear to move because we're shown a rational series of still pictures so rapidly we literally can't tell where one ends and the other kicks off. When you twist the volume knob on the radio, the volume seems to increase smoothly; but volume controls are variable resisters. A piece of wire is wrapped in a coil, and a contact moves up and down the turns of wire to increase or decrease the amount of current. But the least it can adjust the current is by the difference between one wire and the next.

Our perceptions are **analog**—a continuous wavy line of events that move at predictable rates. What's really happening is actually **digital**—represented by discrete, separate, and different numerical values. The light doesn't fade; it's either on or off. You don't move; you're either there or not there. Light can be a wave and it can be a particle. It just depends how you measure it. At the subatomic level, where no one's really figured it all out yet, it might turn out we—everything—are just the vibrations of something we couldn't possibly understand.

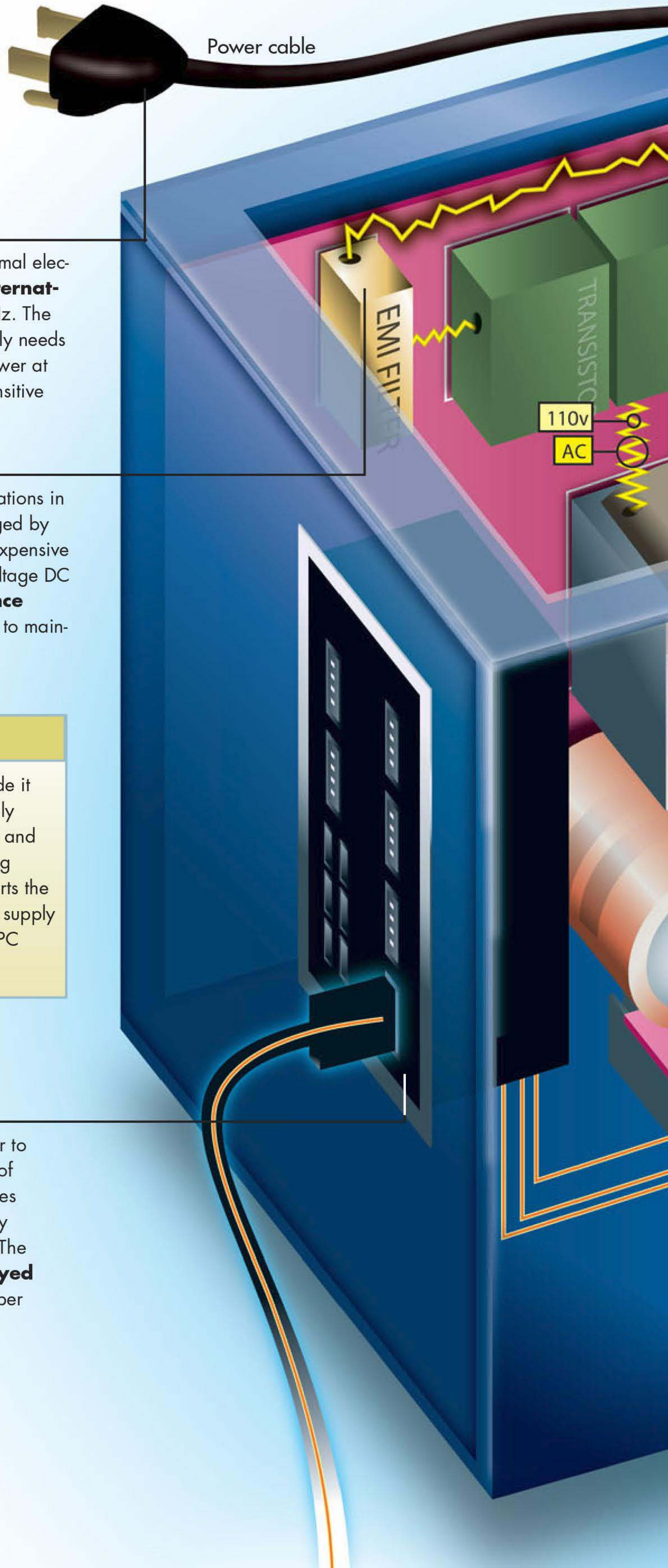
Machines—particularly computers—thrive on digital existences. The metaphor for a computer is a switch. Off means one thing, on means the opposite. 1 and 2, true and false. Black and white. But we've got this analog monkey on our back, so we've had to figure ways to translate the analog values we work with and the digital values of the computer.

The translation you'll encounter most often in this book is an **analog-to-digital converter (ADC)** and the opposite, a **digital-to-analog converter (DAC)**. The idea's simple enough. If you have some current moving through a wire, you just sample it regularly—dozens of times a second at least—and give the amount of current at exactly those moments. And because often we just need to know whether it's yes or no, on or off, we don't really have to be all that exact. If voltage is higher than a certain level, it's on. If it's lower, it's off.

What works so nicely with DACs and ADCs is another part you'll encounter in several computing components, the **charge-coupled device (CCD)**. It consists of an array of sensors made from a material that turns light energy into electrical energy. Light hitting the material must do something with the energy it's carrying, so it's transferred to the electrons in the substance. The electrons start acting like teenagers while their parents are out of town and jump all around, moving from atom to atom: electricity. The stronger the light, the stronger the electrical current, which is sent to an ADC, which turns those different intensities into numbers with which a computer can work.

Inputting data into a computer has, until lately, been slow and deliberate and limited. But we're on the eve of developing computers that are aware of their surroundings with sensors for stimulation we can't detect ourselves. It means that, eventually, input and output will have lost today's meaning. Humans and computers will simply communicate and let us keep our illusions.

# How the Power Supply Works



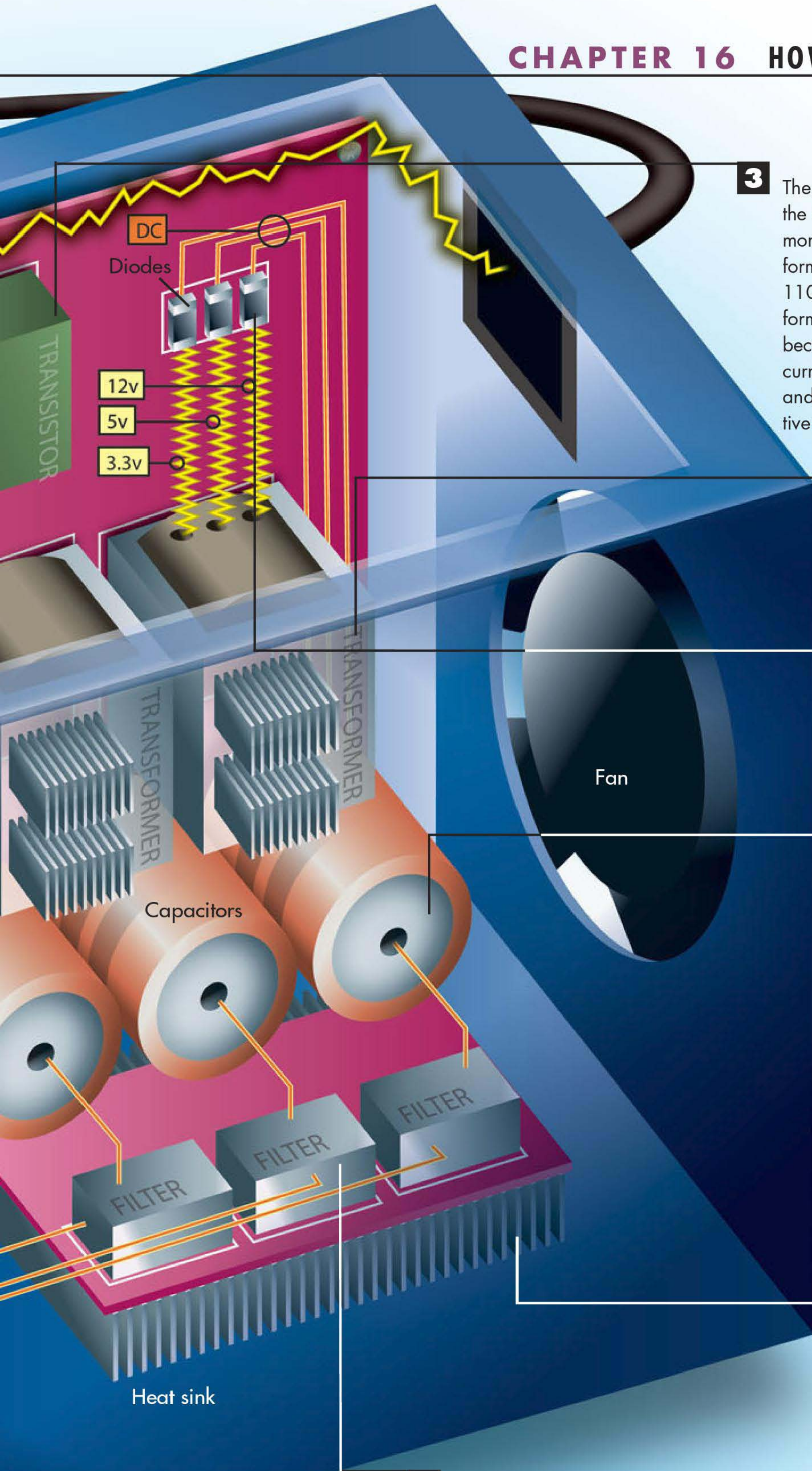

**1** The power supply's power cord plugs into a normal electrical outlet that in the United States provides **alternating current (AC)** power at 110 volts and 60Hz. The 110 volt AC power coming into the power supply needs to be transformed into **direct current (DC)** power at much lower voltages that can be used by the sensitive electronic components inside a PC.

**2** The incoming power is **dirty power**; that is, there are small fluctuations in its current and power. PC components could theoretically be damaged by these variations, and so some power supplies (generally the more expensive ones) filter and condition the power before converting it to lower-voltage DC power. The power flows through an **electromagnetic interference (EMI)** filter to smooth out fluctuations and line-conditioning circuitry to maintain a consistent power level.

### Looking Out for Power No. 1

Power supplies also make sure the PC only starts if the supply can provide it with the proper amount and voltage of electricity. When the power supply starts, it runs a series of internal tests to make sure it is working properly and supplying the right voltage and amount of power. If everything is working properly, it sends a Power Good signal to the PC's motherboard that alerts the motherboard to start up. It is constantly sending that signal. If the power supply stops working properly at any time, it stops sending the signal, and the PC shuts down.

**9** The power is sent to a series of cables that ultimately provide power to the PC's components. Each cable has several wires in it, and each of those wires carries different voltages. Twelve-volt power is for devices such as disk-drive motors, while 3.3-volt and 5-volt power is used by PCI/AGP cards, ISA cards, CPUs, DIMMs, and other components. The cable joins with the PC via cable connectors. Each connector is **keyed** so that it only attaches in a certain way, which ensures that the proper voltage is supplied in the proper place.




**3** The cleaned power goes to a set of **transistors** that convert the 60Hz current to a much higher frequency—one with more cycles per second. This allows small, lightweight transformers to do the work of stepping down the power from 110 volts. If the current stayed at 60Hz, much bulkier transformers would be needed, which would be impractical because they would not fit into a computer. Additionally, AC current at a higher frequency than 60Hz is easier to filter and keep at a constant voltage, which is required by sensitive electronic components inside a PC.

**4** The higher frequency power is sent to **transformers** that step down the voltage from 110 volts to 3.3 volts, 5 volts, and 12 volts, which are the three voltages used by different parts of a PC.

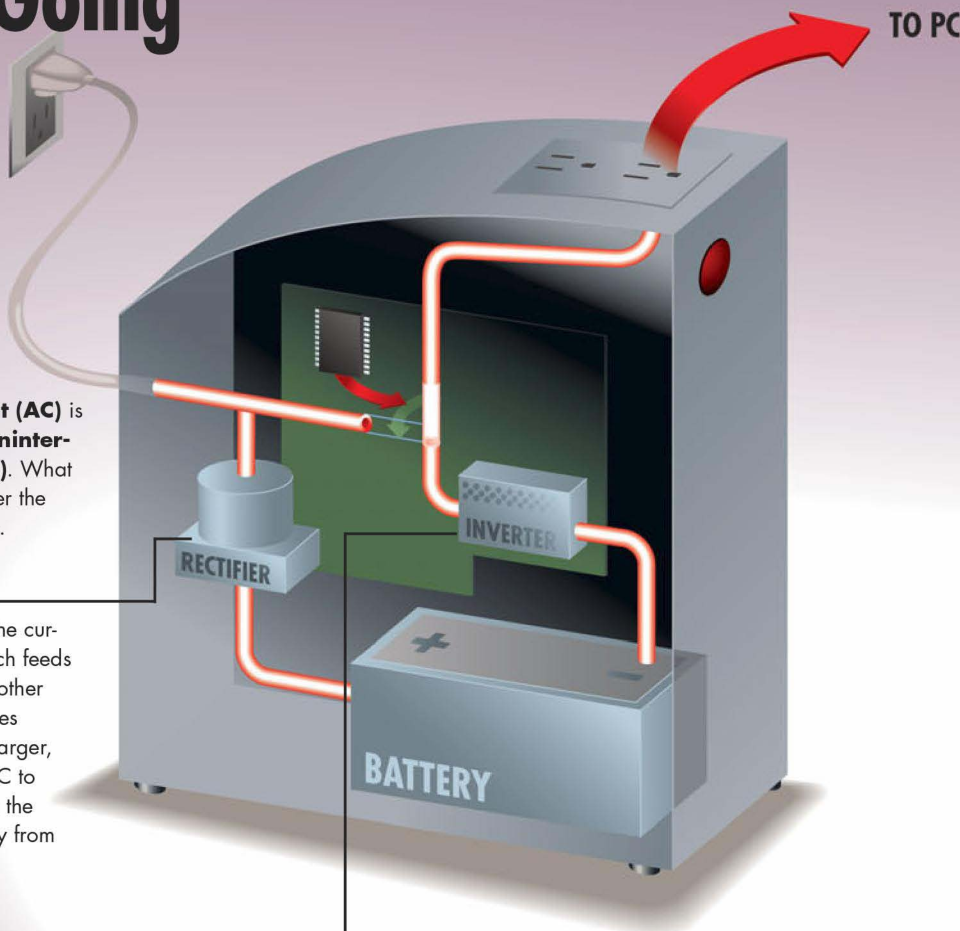
**5** The lower voltage power now goes to **diodes** that **rectify** the power—they change it from AC power to the DC power required by PC components.

**6** Two types of **capacitors** make sure there is always a steady source of power available to a PC's components by storing power and then providing it when necessary. An **input capacitor**, usually the largest capacitor in a power supply, has a reserve of power in it that the power supply draws on when there is a drop of power from the wall outlet, such as when a blow dryer is turned on. Smaller **output capacitors** provide a reservoir of electricity when a computer suddenly needs it, such as when a DVD drive and CD drive simultaneously turn on. Because the capacitors store voltage even when the power supply isn't connected to a wall outlet, they can be exceedingly dangerous (even lethal). You should never open a power supply.

**7** Diodes, transistors, transformers, and capacitors get very hot and could easily burn out if they were not cooled. One or more **heat sinks**, in conjunction with a fan built into the power supply, draw heat away from them and cool them off.

**8** The electricity goes through **filters** that make sure the power is of a constant, non-fluctuating current so electronic components are not damaged by it.

# How a UPS Keeps Your Computer Going



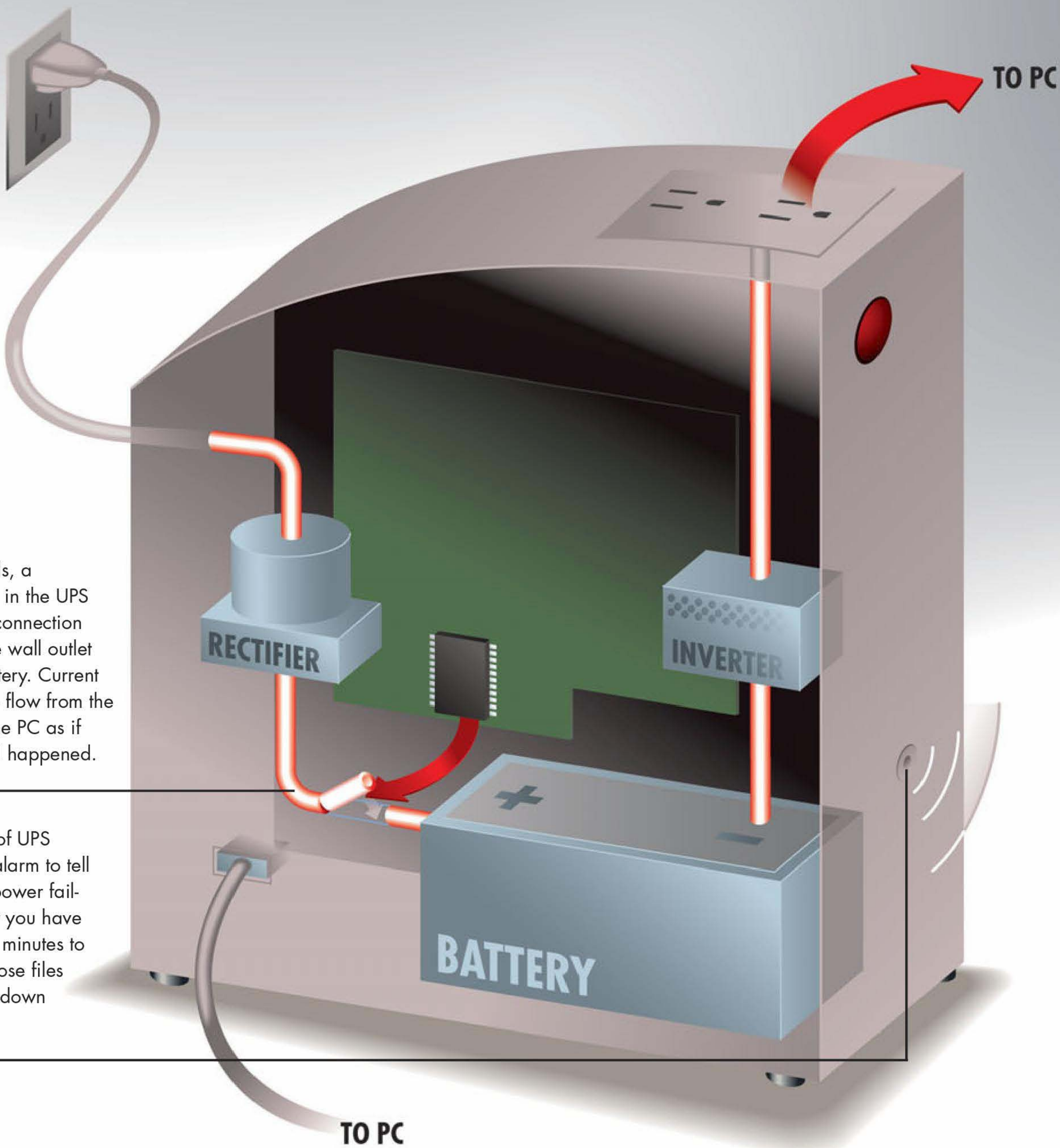

**1** **Alternating electrical current (AC)** is drawn from a wall outlet to the **uninterruptible power supply (UPS)**. What happens then depends on whether the UPS is an offline or online device.

**2** An offline, or passive, UPS splits the current into two branches. One branch feeds power normally to a computer or other peripheral. The other branch passes through a **rectifier**, or battery charger, which changes the current from AC to **direct current (DC)** to replenish the charge that commonly leaks slowly from a battery inside the UPS.

**3** A microprocessor in the offline UPS constantly monitors the main current line for a drop in voltage. If the power fails, within 4–20 milliseconds the microprocessor closes a switch that sends direct current power from the battery through an **inverter**, which turns the battery's DC into the alternating current the computer expects to receive. (High-quality hardware can survive microbreaks in current up to 100 microseconds.)

**4** An online, or serial, UPS feeds all current from the wall outlet through an AC-to-DC converter. From there, the current replenishes a battery and then travels to an inverter before it travels on to the computer as alternating current. Whether there is a power failure or not, the UPS's battery always supplies current to the computer.

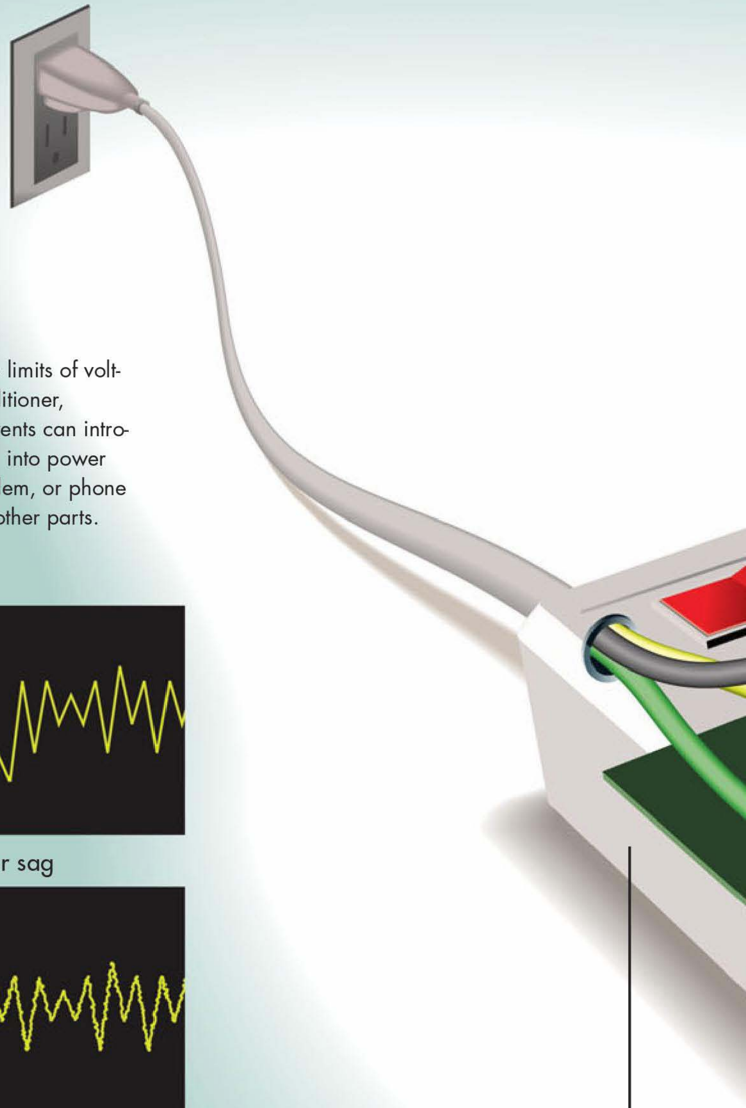
**7** Some UPS units use a serial or USB connection to the PC to launch software that automatically saves files and shuts down the system if no human is handy. In one of the few known cases of electrical irony, when alternating current from the inverter is received by the PC's power supply, the supply converts it to direct current again.



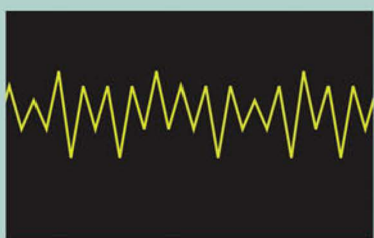

**5** If power fails, a microswitch in the UPS breaks the connection between the wall outlet and the battery. Current continues to flow from the battery to the PC as if nothing had happened.

**6** Either type of UPS sounds an alarm to tell you of the power failure. Usually you have at least five minutes to save and close files and to shut down the PC.

# How Surge Protectors Work



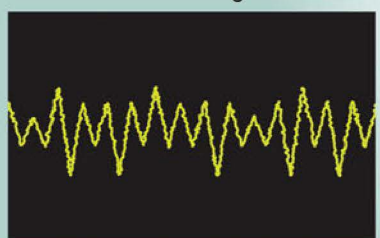
**1** Under ideal circumstances, electrical current would stay within close limits of voltage, frequency, and other characteristics. But turning on an air-conditioner, nearby electrical lines, close lightning strikes, and other electrical events can introduce **power sags**, **power surges**, or general **electrical noise** into power and telephone lines. These disturbances can cause a computer, modem, or phone to malfunction. They can crash a PC or destroy its power supply or other parts.



Ideal current

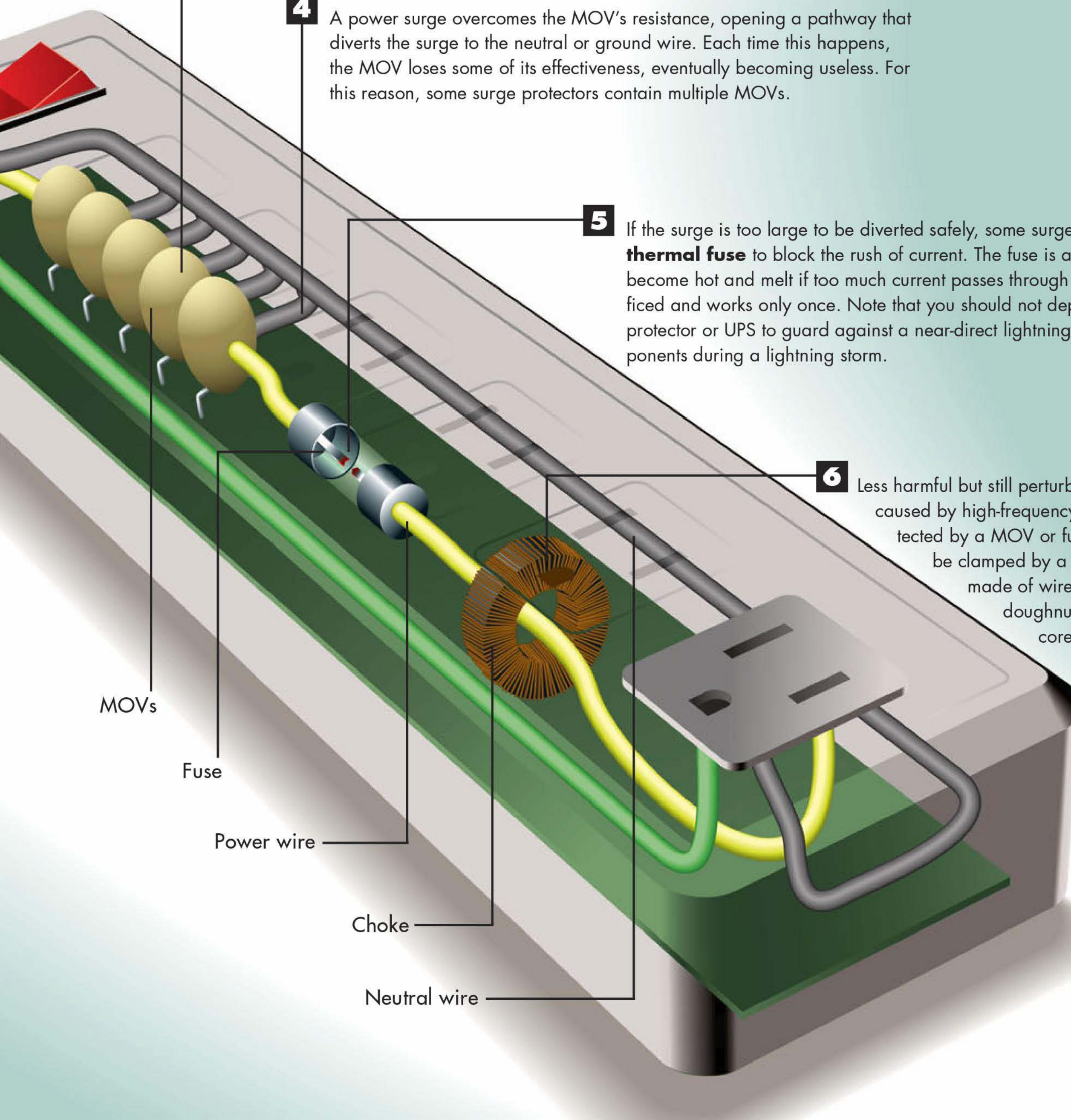
Power sag

Power surge



Electrical noise

**2** Most UPS devices also provide some form of **line conditioning**, which tries to compensate for the irregularities in current. The act of converting current from AC to DC and back again helps eliminate some irregularities. Or a **surge protector**, an electrical extension cord that includes components to tackle electrical distortions, provides basic line conditioning.




**3** The first line of defense against a **power surge**—a sudden spike in the voltage—is the **shunt mode**. It uses a **metal-oxide varistor (MOV)** between the power line and its neutral or ground line. The MOV consists of metal-oxide material, such as zinc oxide, which transmits electricity separated by semiconductors that don't carry current unless the voltage reaches a certain level.

**4** A power surge overcomes the MOV's resistance, opening a pathway that diverts the surge to the neutral or ground wire. Each time this happens, the MOV loses some of its effectiveness, eventually becoming useless. For this reason, some surge protectors contain multiple MOVs.

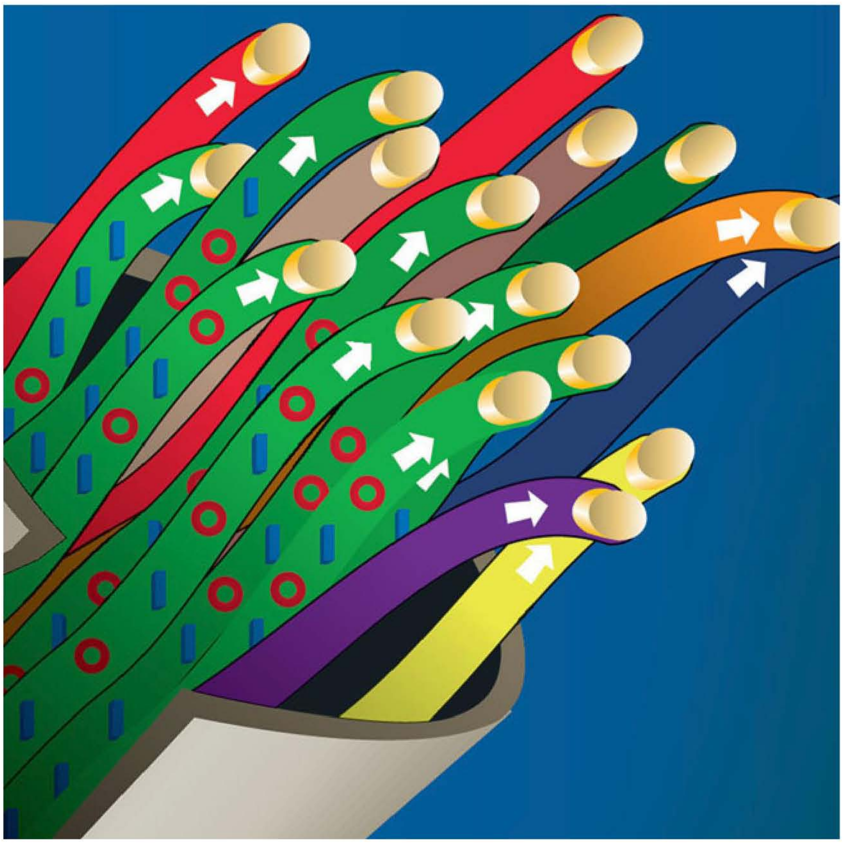
**5** If the surge is too large to be diverted safely, some surge protectors use a **thermal fuse** to block the rush of current. The fuse is a resistor designed to become hot and melt if too much current passes through it. The fuse is sacrificed and works only once. Note that you should not depend on any surge protector or UPS to guard against a near-direct lightning strike; unplug components during a lightning storm.

**6** Less harmful but still perturbing line noise—often caused by high-frequency radio waves—is undetected by a MOV or fuse. But some of it might be clamped by a **toroidal choke coil** made of wire wrapped around a doughnut-shaped magnetic core. Current on the way to the PC passes through the hole in the core. Fluctuations in the current create another current in the choke's coiled wire. That current, in turn, creates an electromagnetic field that opposes the noise, smoothing out the current.

CHAPTER

17

# How Serial Ports Triumph



**THIS** book used to have one of those explanations that made so much sense, that clicked with our common perceptions so precisely, that they could only be called elegant. The explanation went like this:

Computers have two basic types of connections for sending data from here to there: serial connections and parallel connections. Serial connections were used with peripherals that are inherently slow, such as modems, which are themselves serial devices. Parallel connections invariably led to components such as hard drives, where components moved at breath-taking speeds to handle the enormous quantities of data. The reason why, so the explanation went, was that serial connections could only send bits of data down a single wire one at a time. It was serial in the same sense of a serial killer, who murders only one victim at a time. Parallel connections were the mass murderers of communications, moving several bits *at the same time*, down several wires that were laid out, parallel to each other, in a cable. The explanation encouraged the reader to think of serial connections as soldiers marching in a single line. Parallel cables were like soldiers marching eight or more abreast. Which formation would get an army of data into the fray faster? Why, it was obvious.

Until the soldiers marching in formation began stumbling all over themselves.

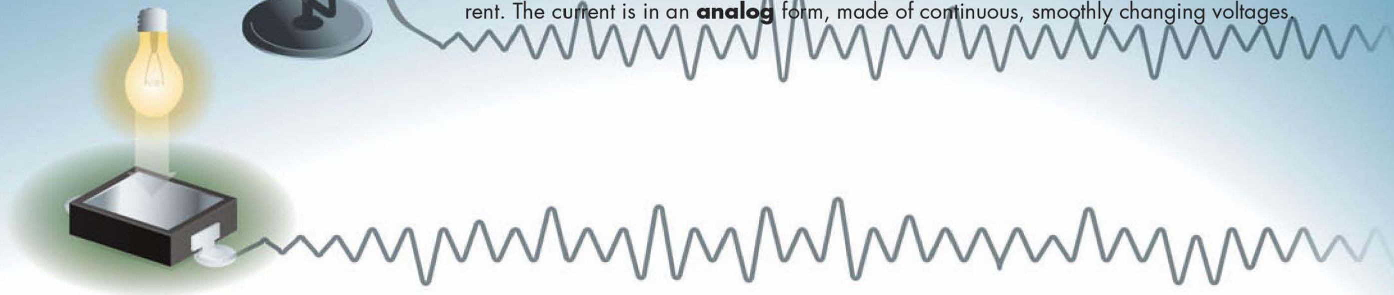
When a device is reading bits of data, it's actually detecting a change in the voltage level of voltage on a wire or circuit. A high voltage—usually 3 or 5 volts in a computer—stands for a 1 bit; a low voltage (or no current) represents a 0 bit. Such changes aren't instantaneous. It takes a few milliseconds for the voltage to climb to the high point or to dissipate down to a low voltage. Then it must remain that way long enough for the device to be sure it's not just some momentary spike or drop. So, if you built a device that does a better job of recognizing differences in voltage, instead of 5 volts, the circuits can use 3 volts, which takes less time to generate.

The problem with parallel connections is that it expects all of its signals coming in on separate wires to arrive at precisely the same time. Push the speed of those signals by making them shorter or using a lower voltage, and parallel ports generate constant errors. The signals don't arrive at the same time, and if there's an error on one line, the signals on all lines would have to be resent.

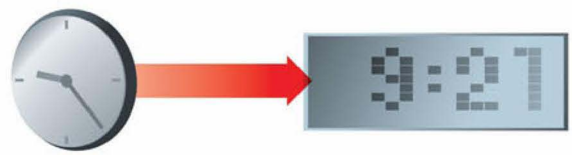
A serial connection, however, can easily ask for a resend of an erroneous signal. It works because serial bits are sent as groups inside a packet that contains other helpful information about the data, including where the packet's bits belong in the sequence of signals. And, as it turns out, sending bits one after the other is faster than trying to make a lot of parallel signals march in precision.

# Analog and Digital Converters

**1** Computers extract information from the world around them using devices such as microphones and photosensitive diodes to detect changes in energy levels. Inside a microphone, the energy from sound waves vibrates metal plates, which changes the voltage in an electrical current passing through the plates. The photodiodes are made with a material that absorbs light's energy and converts it into electrical energy. The stronger the light hitting the diode, the stronger the electrical current. The current is in an **analog** form, made of continuous, smoothly changing voltages.

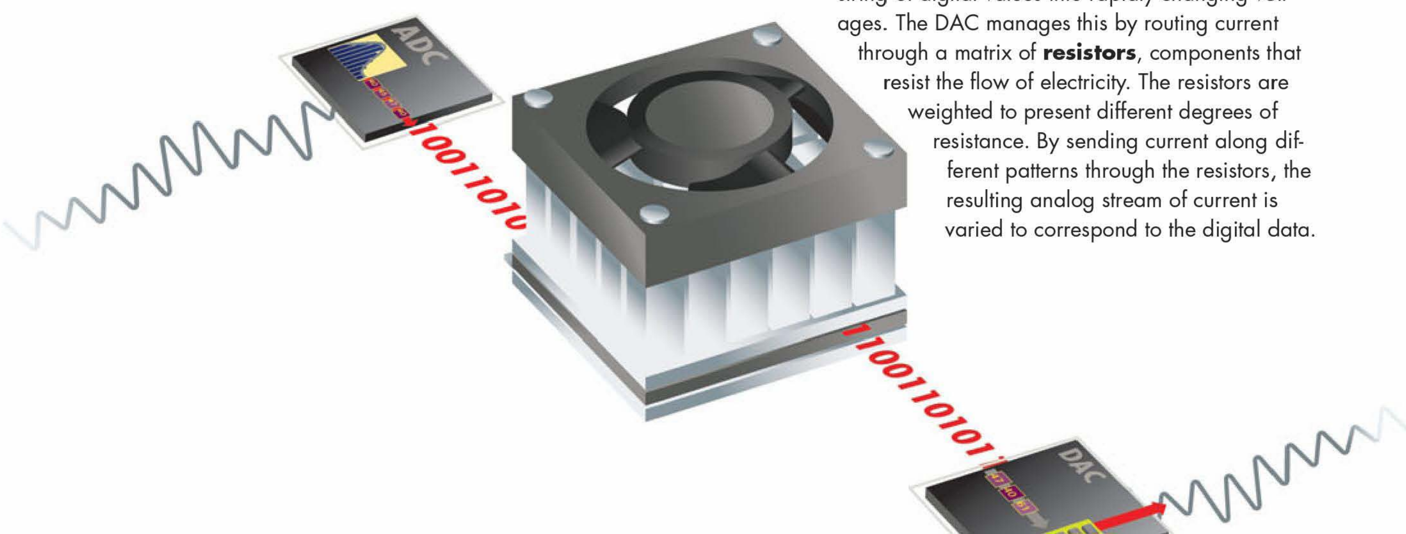


**2** Computers are designed to manipulate **digital** data—specific values are expressed in real numbers. A computer must convert all analog input it receives into digital values before the PC can use it. Conversely, most of the results of the computer's digital manipulation must be converted back to analog when it becomes **output**.

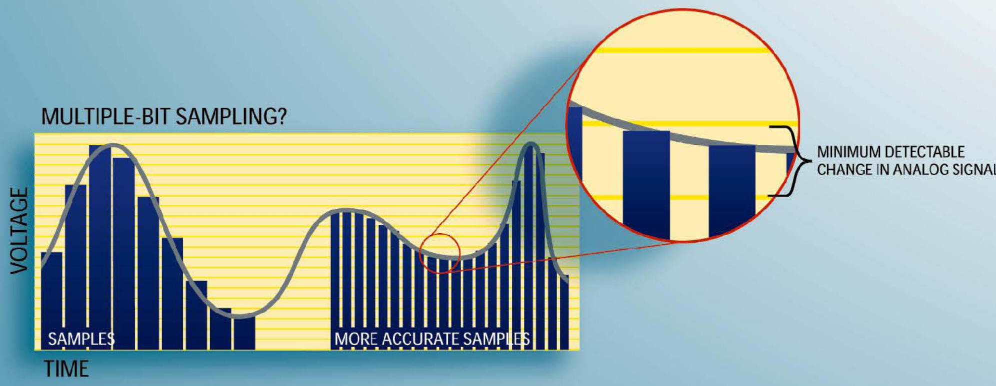



**3** A microchip called an **analog-to-digital converter (ADC)** constantly samples an analog signal in the form of a wavering electrical current. Each time the current is measured, the ADC generates a number that represents the analog value of the current at that moment.

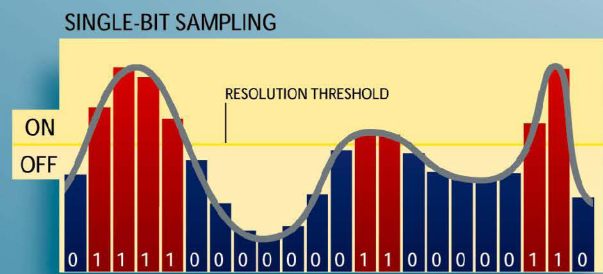
**4** On the output end, analog currents are needed by most monitors, sound speakers, and modems. A **digital-to-analog converter (DAC)** changes a string of digital values into rapidly changing voltages. The DAC manages this by routing current through a matrix of **resistors**, components that resist the flow of electricity. The resistors are weighted to present different degrees of resistance. By sending current along different patterns through the resistors, the resulting analog stream of current is varied to correspond to the digital data.

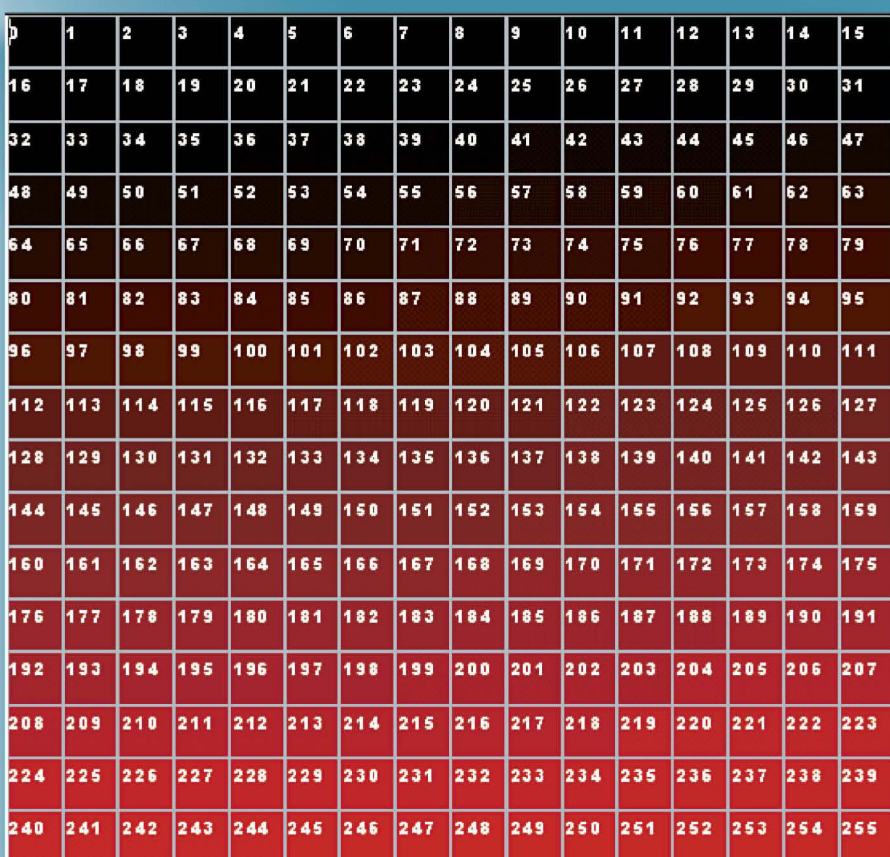


**5** The precision of an ADC/DAC conversion is affected by how often the analog signal is sampled. Samples made frequently detect finer changes in the analog signal. The precision is also affected by the sensitivity of the converters. Two parts of the analog signal can differ so slightly that an ADC cannot detect a difference, so it assigns both parts the same digital value.




**6** **Resolution** is the range of analog values an ADC or DAC can handle and, consequently, how much information a device can hold for any one sample. Resolution depends on the number of bits the devices can devote to the digital translation of the analog values. If the DAC has only one bit to represent any one sample of the analog signal, it can only show whether the analog signal is on, representing white (1), or off and black (0). One-bit samples are sufficient for scanning printed text for conversion into text you can edit with your PC.



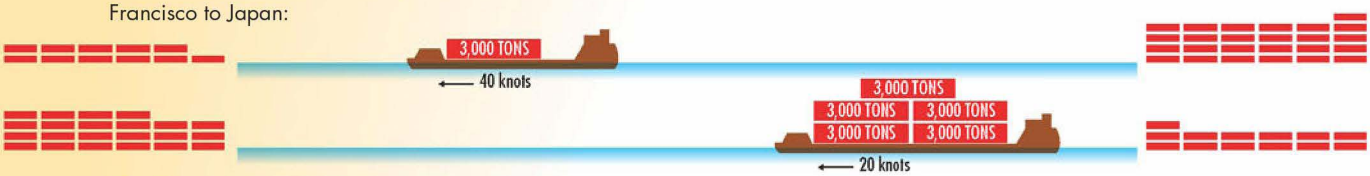

**7** However, if a DAC has 16 bits to represent each of the colors—red, blue, and green—that make up a pixel, those 16 bits can express any of 256 shades of each of the colors. The array of red shown here demonstrates that the higher the number, the more saturated the red is. A 16-bit value of 0000000000000000 (0 decimal) is completely black and is one of the 256 possible values. A 16-bit value of 1111111111111111 (255 decimal) would be as intensely red as the monitor can produce. Combined with 256 shades of blue and green, 16-bit, or high color, produces a total of 16,777,216 colors. The array of colored squares here shows each of the 256 values for red that a 16-bit display produces. For most adjacent values, the variation is smaller than the human eye can detect.

# How Bandwidth Moves Your Data

**1** Information—data—moves from one place to another riding on the crests and valleys of waves called **carriers**. Carriers are simple, pure waves of sound or electromagnetism, which includes light, heat, radio and TV transmissions, X-rays, and every other type of radiation. (See the spread titled "How Electromagnetism Reacts with Matter" in Chapter 10.)

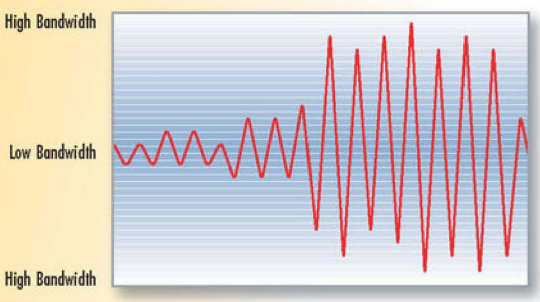
**2** People and machines generate more complex, but weaker, waves that represent words, images, music, and electronic data. These **modulator waves** superimpose their wave patterns on the simpler carrier waves. The result is an entirely new **waveform**, similar to when two ripples on a pond collide, creating an entirely new set of ripples.

**3** How much data a carrier moves from *here* to *there* in one second is that waveform's **bandwidth**. Although we talk of one bandwidth being faster than another, what we're really measuring is **capacity**. For example, imagine two ships traveling from San Francisco to Japan:




- Ship A carries 3,000 tons traveling at a steady 40 knots.
- Ship B is half as fast, traveling at 20 knots.

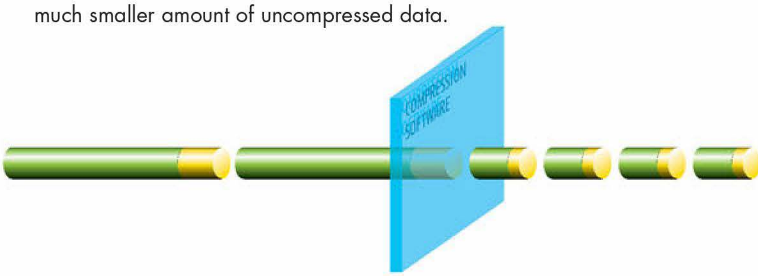
However, Ship B carries 15,000 tons, so it has the greater bandwidth. If each of the ships had to transport 100,000 tons, it would take Ship A 366 days to move all the material. Ship B would finish the job sooner, in 150 days, not because Ship B is faster—it's not—but because it has a broader bandwidth (it carries more).

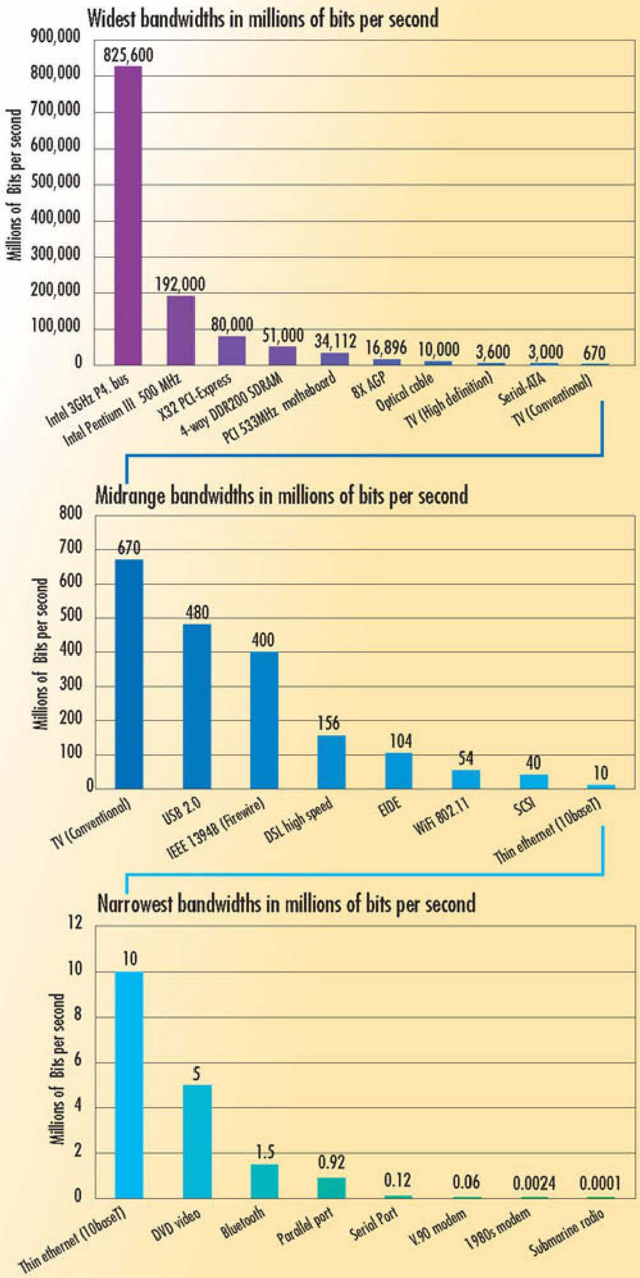



**Increase the frequency.** Each cycle of the carrier wave is a new opportunity for data to merge with the carrier. The higher the frequency of the waves, the greater its capacity to carry data. We measure frequencies in **hertz**, which is one oscillation a second.

**4** Each part of a PC—its processor, disk drives, system bus, video card—has a bandwidth. Any one of these **channels** can become a system bottleneck if its bandwidth doesn't keep pace with the bandwidths of other components. There are several ways to boost bandwidth:

**Compress the data.** Zipped files and MP3 songs allow an Internet connection to send more information in the same time it would take to send a much smaller amount of uncompressed data.

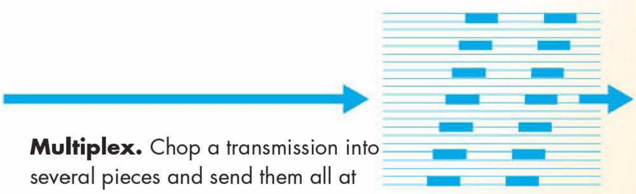





**Cache data.** The device saves copies of frequently repeated data in a special pool of memory called a **cache**. The next time the same data is needed, it can be fetched quickly from the cache.

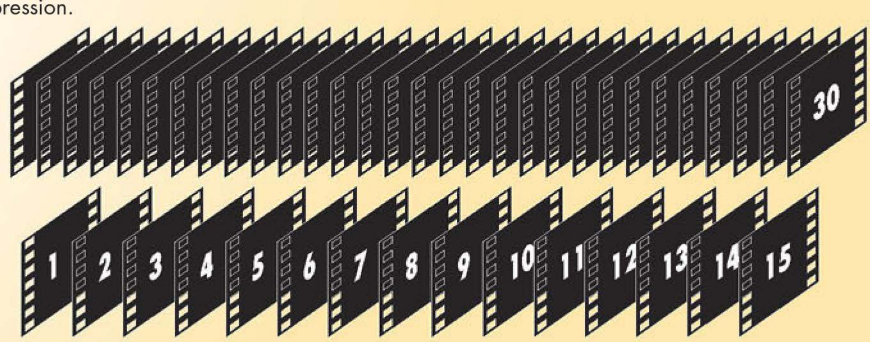
**Reduce latency.** The theoretical bandwidth, or **throughput**, is rarely realized in reality. As much as 60 percent of a transmission's potential bandwidth is lost when many of the bits being transmitted don't represent data. Instead, they are used for addressing, identification, error-checking, and other necessary chores that are collectively called **latency**. Using improved materials and more sensitive signal detectors can reduce latency.

**Prefetch data.** A device makes an educated guess as to what data it will have to handle next and **prefetches** that data so it's already waiting to be processed when its time comes. Doctors are prefetching patients when they keep their waiting rooms stocked with more patients than they can treat at one time.



**Multiplex.** Chop a transmission into several pieces and send them all at the same time on different channels, such as different radio frequencies. It's like a grocery store with two check-out counters or 20.

**5** We most often hear of bandwidth in connection with Internet connections, particularly **broadband**, which is a high-capacity line, such as DSL and cable. But anything that moves some sort of data has a bandwidth. The charts shown here give you an idea of the relative bandwidths of different means of transmitting data. The band capacities here don't take into consideration latency or methods of overcoming latency, such as compression.



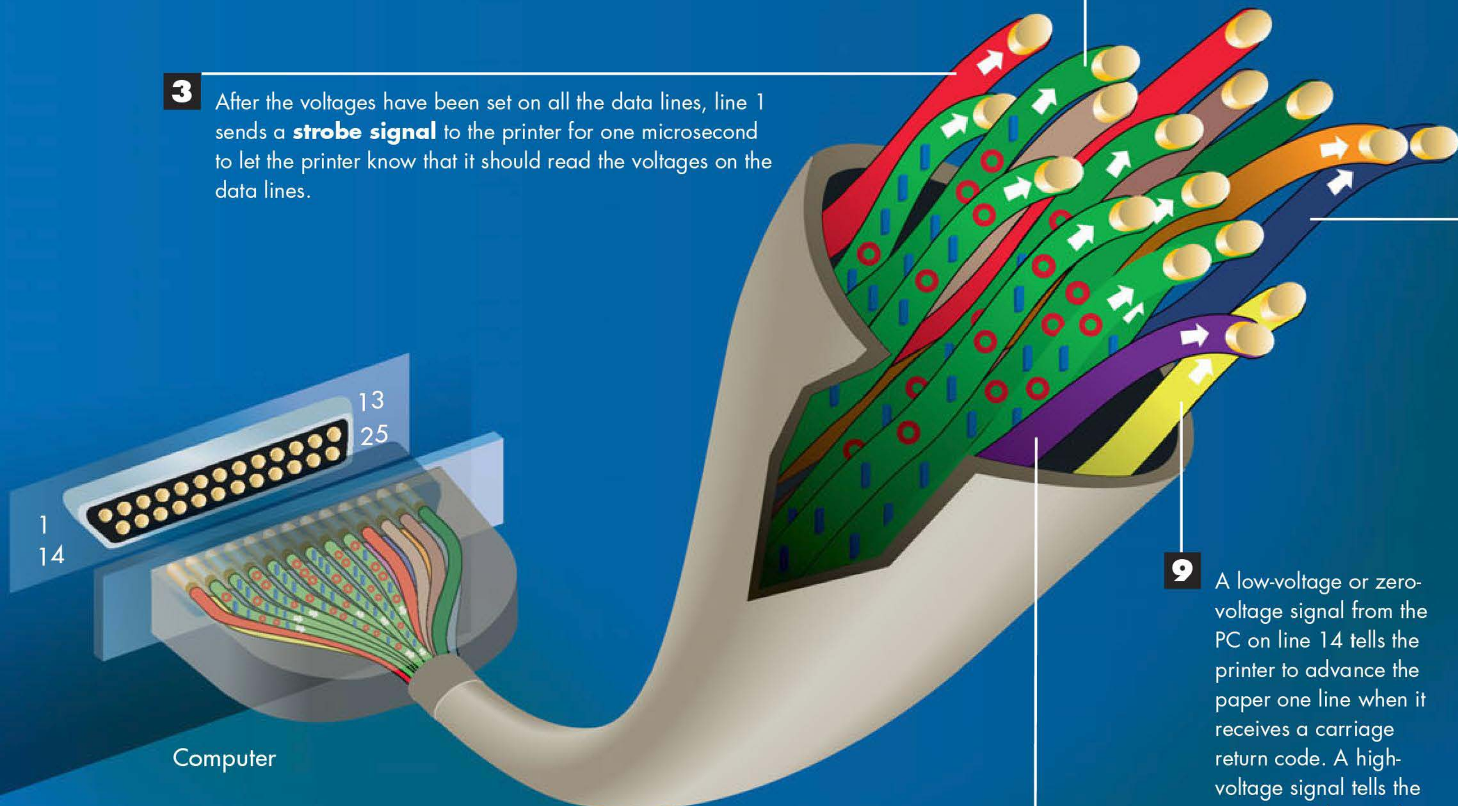

**Send less data.** If you don't have the bandwidth to send high-resolution color images, save them as black-and-white at a lower resolution, which requires fewer bits to record the image. If you can't send 30 frames a second of video, send 15 frames.

# How a Parallel Port Works

**1** A signal to the PC on line 13—called the **select line**—from the peripheral, usually a printer, tells the computer that the printer is online and ready to receive data.

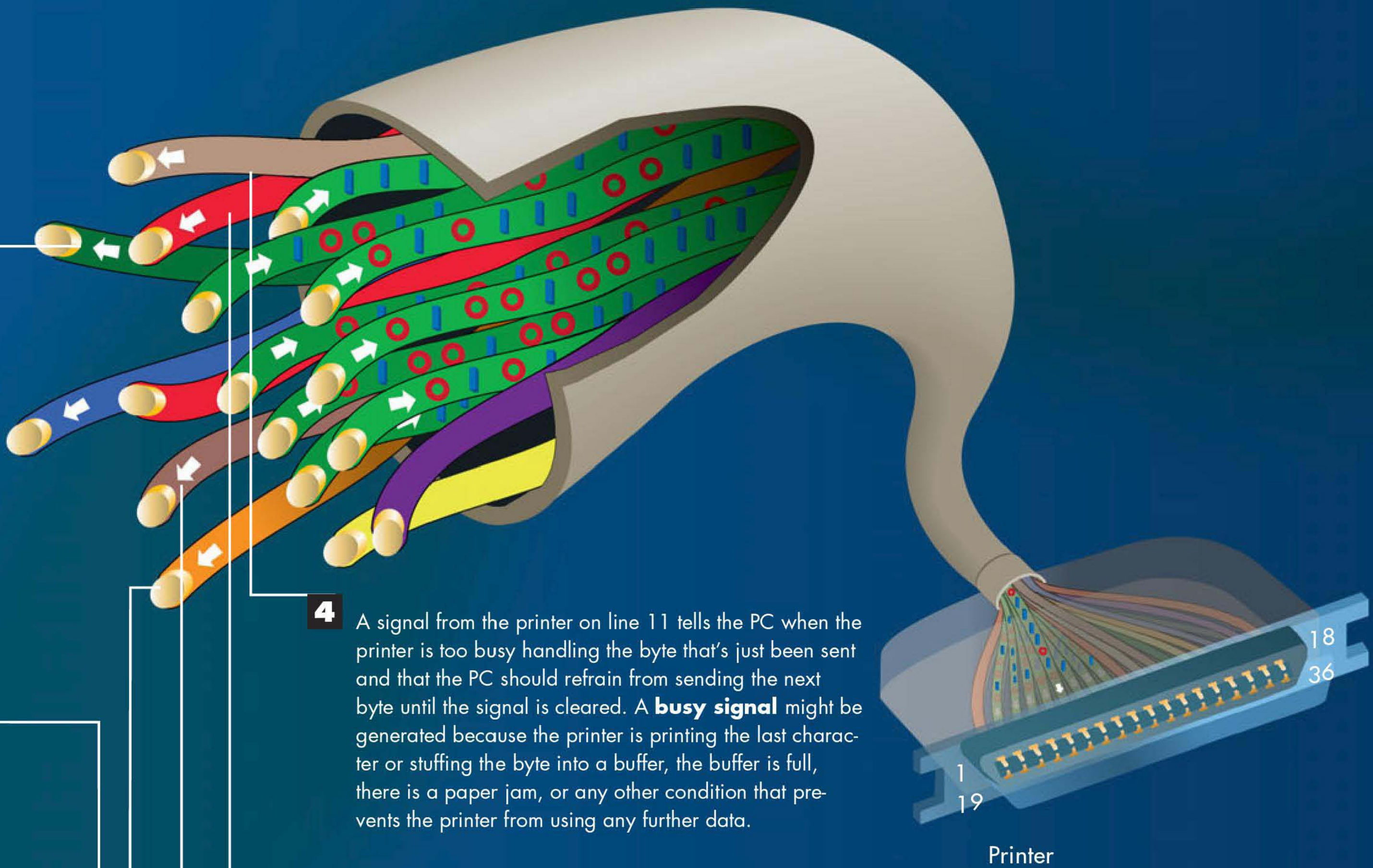
**2** Data is loaded on lines 2 through 9, shown here as green, in the form of a "high" voltage signal—actually about five volts—to signify a 1, or a low, nearly zero voltage signal to signify a 0.

**3** After the voltages have been set on all the data lines, line 1 sends a **strobe signal** to the printer for one microsecond to let the printer know that it should read the voltages on the data lines.




**9** A low-voltage or zero-voltage signal from the PC on line 14 tells the printer to advance the paper one line when it receives a carriage return code. A high-voltage signal tells the printer to advance the paper one line only when it receives a line-advance code from the printer.

**10** A signal from the PC on line 17 tells the printer not to accept data. This line is used only with some printers, which are designed to be switched on and off by the PC. Lines 18 through 25 are simply ground lines.




**4** A signal from the printer on line 11 tells the PC when the printer is too busy handling the byte that's just been sent and that the PC should refrain from sending the next byte until the signal is cleared. A **busy signal** might be generated because the printer is printing the last character or stuffing the byte into a buffer, the buffer is full, there is a paper jam, or any other condition that prevents the printer from using any further data.

**5** A signal from the printer on line 10 **acknowledges** receiving the data sent on lines 2 through 9 and tells the PC that the printer is ready to receive another character.

**6** Line 12 sends a signal from the printer to the PC if the printer runs out of paper.

**7** The printer uses line 15 to tell the PC some error condition exists, such as a jammed print head or an open panel. However, it doesn't specify the nature of the error.

**8** A signal from the PC on line 16 causes the printer to **reset** itself to its original state–the same as if the printer were turned off and on.

### Parallel Soldiers

Think of parallel communications as being like a platoon of soldiers marching eight abreast. Draw a line in the ground in front of the soldiers, and eight soldiers will cross it simultaneously, followed by the eight soldiers behind them. A battalion marching in this manner could cross the line in about 10 seconds.

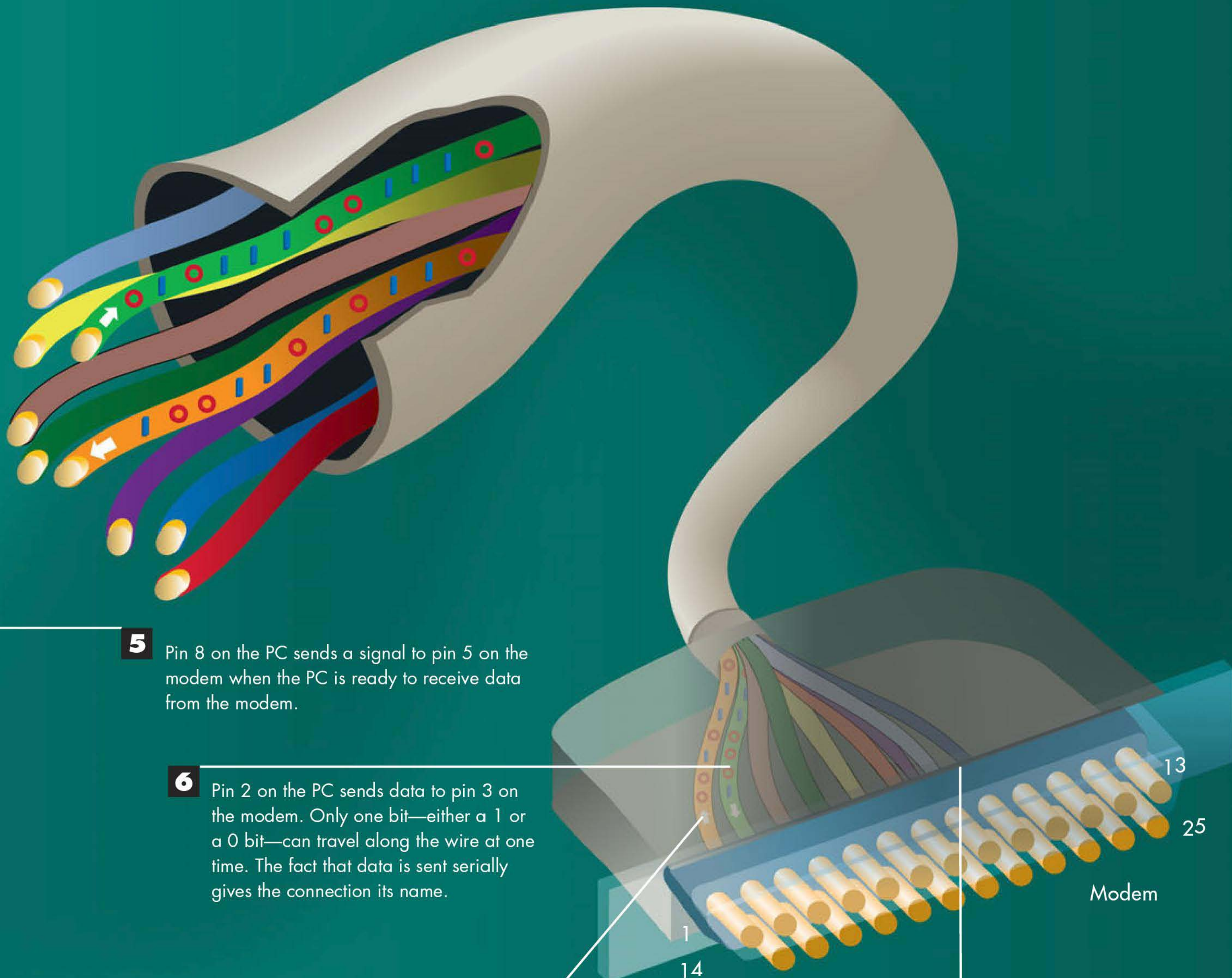
# How a Serial Port Works

**1** Because it is one of the most common uses of a serial port, we'll use a dial-up modem connection to explain how the port works. (See Chapter 25, "How PCs Connect to the Internet," for more information on modems.) Pin 1 and pin 5 on the computer's port connect, respectively, to pin 8 and pin 7 on the modem port. Pins 1 and 8 share a **common ground** connection. Pins 5 and 7 let the PC detect a phone-line signal.

**2** Pin 6—the same on both ends—sends a signal that data is ready to be sent (**data ready**).

**3** Pin 4 on the PC connects to pin 20 on the modem. It signals that the PC is working properly to receive data.

**4** Pin 7 on the PC connects to pin 4 on the modem. The modem puts a voltage on this line to create a request, a signal to the PC that the modem is ready to receive data.




**5** Pin 8 on the PC sends a signal to pin 5 on the modem when the PC is ready to receive data from the modem.

**6** Pin 2 on the PC sends data to pin 3 on the modem. Only one bit—either a 1 or a 0 bit—can travel along the wire at one time. The fact that data is sent serially gives the connection its name.

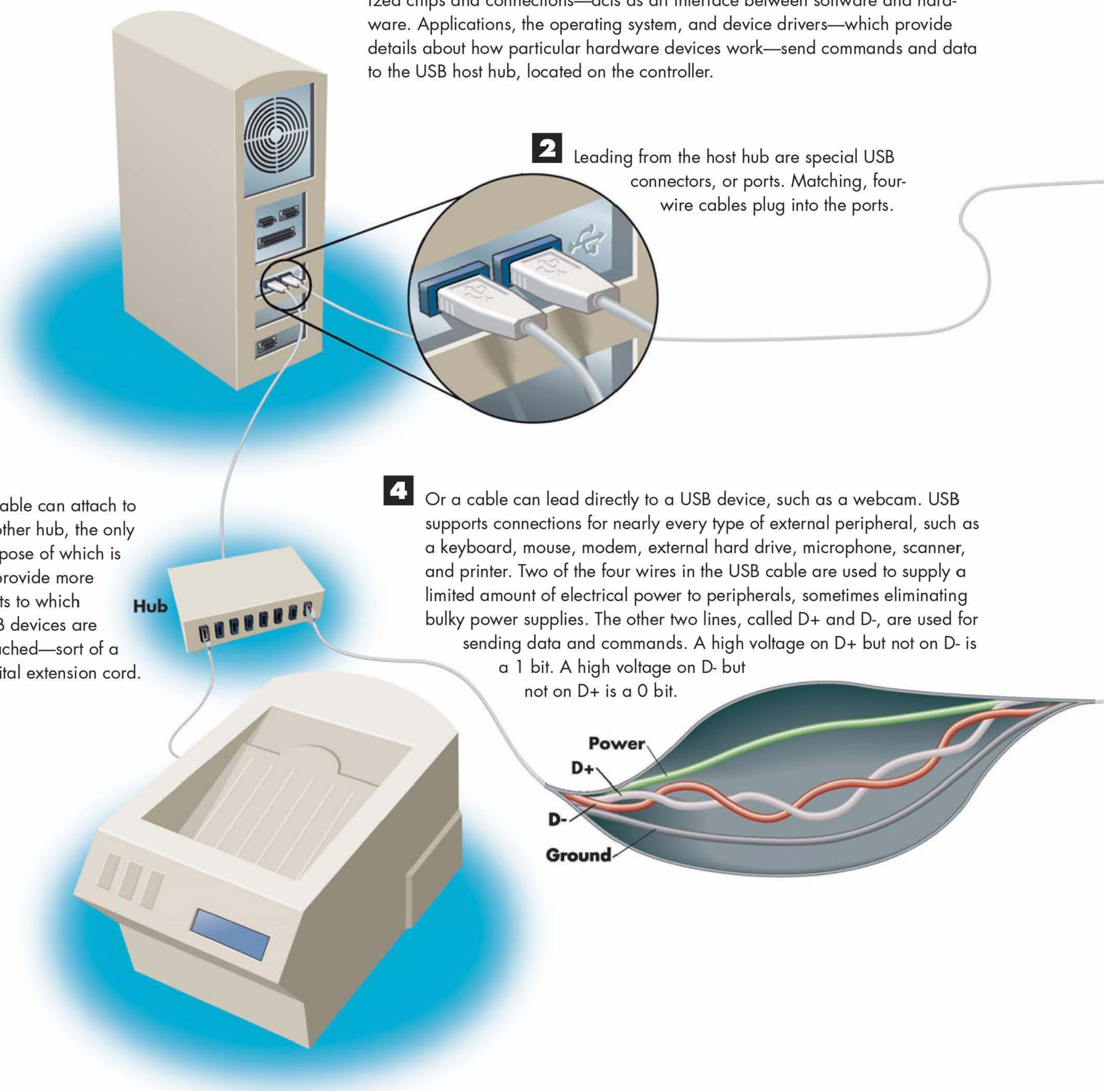
**7** Pin 3 on the PC receives data from pin 2 on the modem. Again, the bits can move through the wire only one bit at a time.

**8** Pin 9 on the PC connects to pin 22 on the modem to detect a telephone ring.

**Soldiers All in a Row**

A serial connection is comparable to soldiers lined up in a single row. Only one of them at a time would be able to cross a line drawn on the ground. It would take more than a minute for all the soldiers in a battalion to cross the line serially.

# How the Universal Serial Bus (USB) Works




**1** Inside the PC, a **universal serial bus (USB) controller**—a set of specialized chips and connections—acts as an interface between software and hardware. Applications, the operating system, and device drivers—which provide details about how particular hardware devices work—send commands and data to the USB host hub, located on the controller.

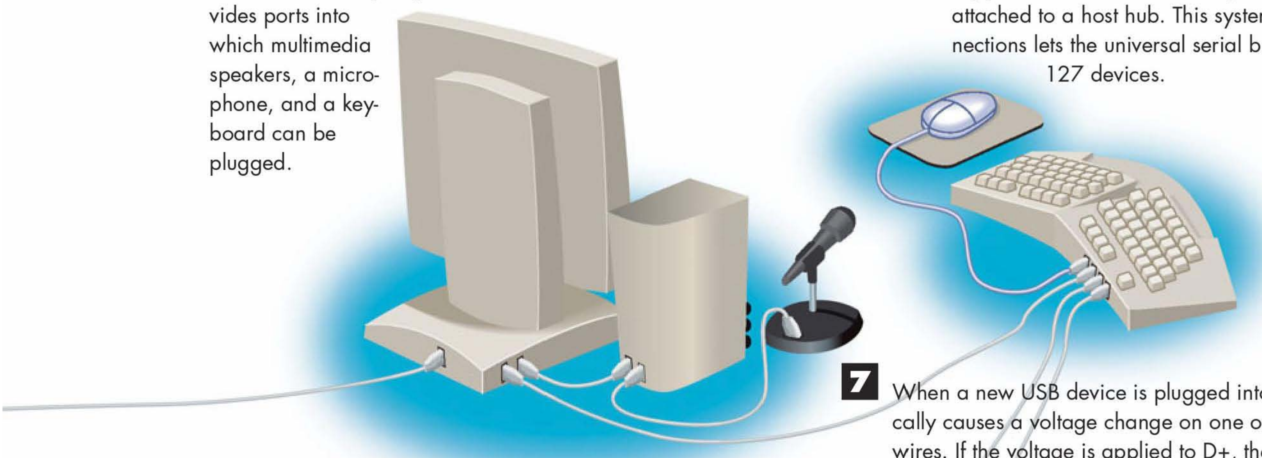
**2** Leading from the host hub are special USB connectors, or ports. Matching, four-wire cables plug into the ports.

**3** A cable can attach to another hub, the only purpose of which is to provide more ports to which USB devices are attached—sort of a digital extension cord.

**4** Or a cable can lead directly to a USB device, such as a webcam. USB supports connections for nearly every type of external peripheral, such as a keyboard, mouse, modem, external hard drive, microphone, scanner, and printer. Two of the four wires in the USB cable are used to supply a limited amount of electrical power to peripherals, sometimes eliminating bulky power supplies. The other two lines, called D+ and D-, are used for sending data and commands. A high voltage on D+ but not on D- is a 1 bit. A high voltage on D- but not on D+ is a 0 bit.

**5** Any USB device can also include a hub, so that a monitor, for example, provides ports into which multimedia speakers, a microphone, and a keyboard can be plugged.

**6** These devices can, in turn, provide ports for further USB hardware. For example, a mouse and a digitizing pen could attach to the keyboard, which is attached to a host hub. This system of branching connections lets the universal serial bus handle up to 127 devices.

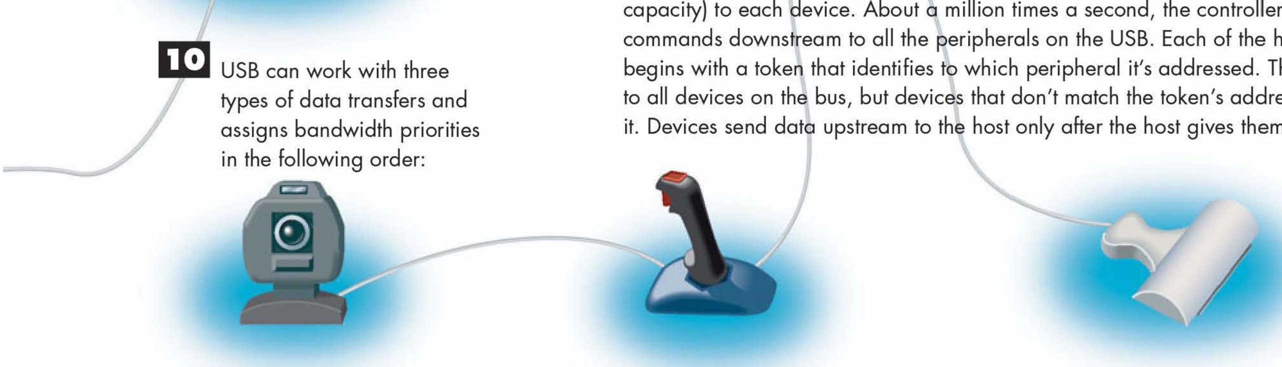


**7** When a new USB device is plugged into a port, it automatically causes a voltage change on one of the two data wires. If the voltage is applied to D+, the peripheral is saying it's a high-speed device, capable of sending 12 megabits a second (Mbps), used for monitors, scanners, printers, and other devices that send a high volume of data. A voltage on D- indicates it can get by with a slow transfer speed of 1.5Mbps, say, for a keyboard or mouse. (A conventional serial port, in comparison, sends only 100 kilobits a second; a parallel port about 2.5Mbps. Hi-Speed USB sends 480Mbps.)

**8** Working with similar Plug and Play technology that allows automatic configuration of internal PC components, the USB host controller tells the new device to identify itself, finds out what it requires for sending and receiving data, and assigns the device an identification number.


**9** Now that the new device is an official member of the bus, it takes its place as the host controller polls the devices—to issue commands, to ask whether the device is ready to send or receive data, and to apportion chunks of the bus's bandwidth (data-transmitting capacity) to each device. About a million times a second, the controller sends queries or commands downstream to all the peripherals on the USB. Each of the host's messages begins with a token that identifies to which peripheral it's addressed. The message goes to all devices on the bus, but devices that don't match the token's address simply ignore it. Devices send data upstream to the host only after the host gives them permission.

**10** USB can work with three types of data transfers and assigns bandwidth priorities in the following order:

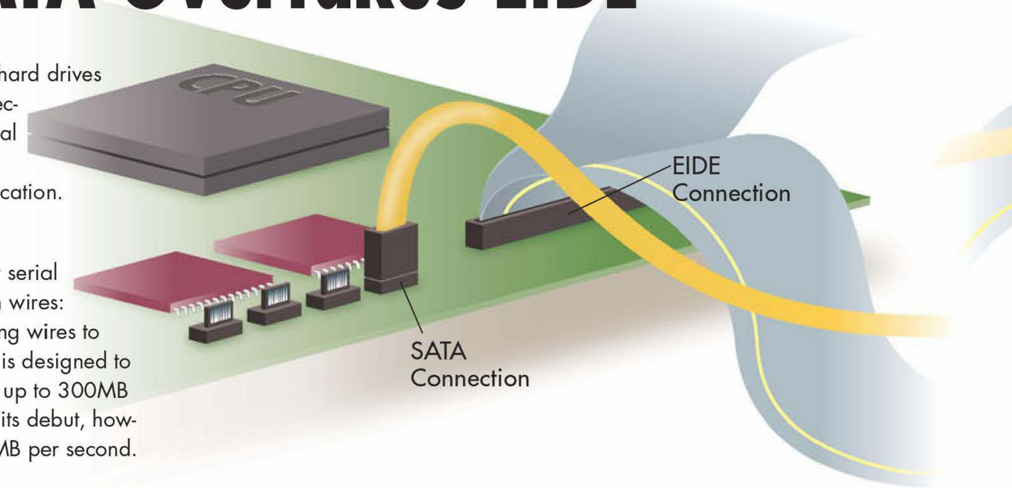


**Highest priority**
**Isochronous**, or real-time, where there can be no interruption in the flow of data, such as video or sound.

**Second highest priority**
**Interrupt transfers**, which occur only when a device, such as a keyboard or joystick, generates an occasional interrupt signal to get the processor's attention.

**When time permits priority**
**Bulk transfers** of data for printers, scanners, and digital cameras, in which there's a lot of data to send but no particular hurry to get it there.

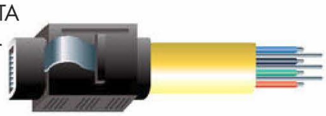
# How Serial ATA Overtakes EIDE



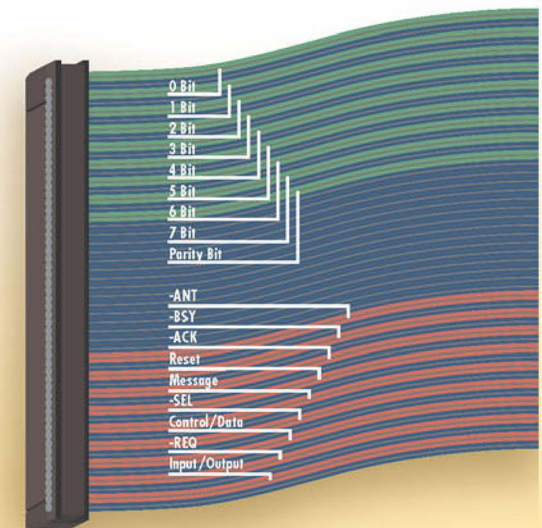

1. The EIDE/ATA connection standard for hard drives tops out at transfer rates of 133MB a second. Above that, **crosstalk**, or electrical interference, from their 40-wire parallel cables drowns out meaningful communication.

2. Its replacement is SATA, which stands for serial AT-attachment. Its cables have only seven wires: four for carrying data and three grounding wires to dampen any crosstalk. The arrangement is designed to allow the cables to have a bandwidth of up to 300MB per second or 3 gigabits per second. At its debut, however, the technology was limited to 150MB per second.

3. With a planned **port multiplier**, a single SATA cable can lead to 15 hard drives that are hot-swappable so they can be unplugged and inserted while the computer is running. Each drive has its own channel and recognizes commands from the motherboard only if the signals identify it as the recipient. The arrangement eliminates the need to designate drives as master and slave. Although SATA does support optical drives, only a handful of these drives are on the market because they do not benefit from SATA's increased bandwidth.

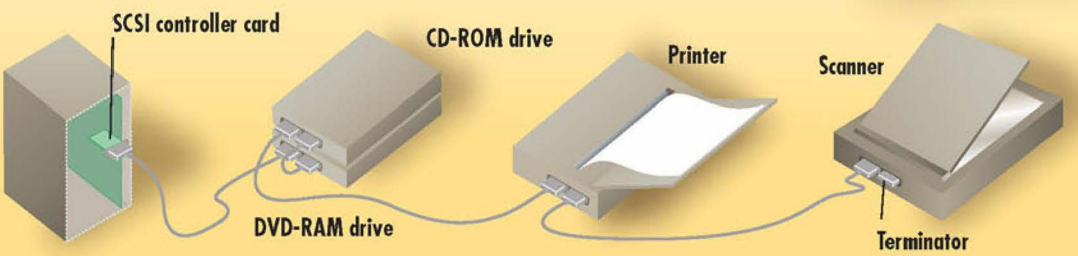


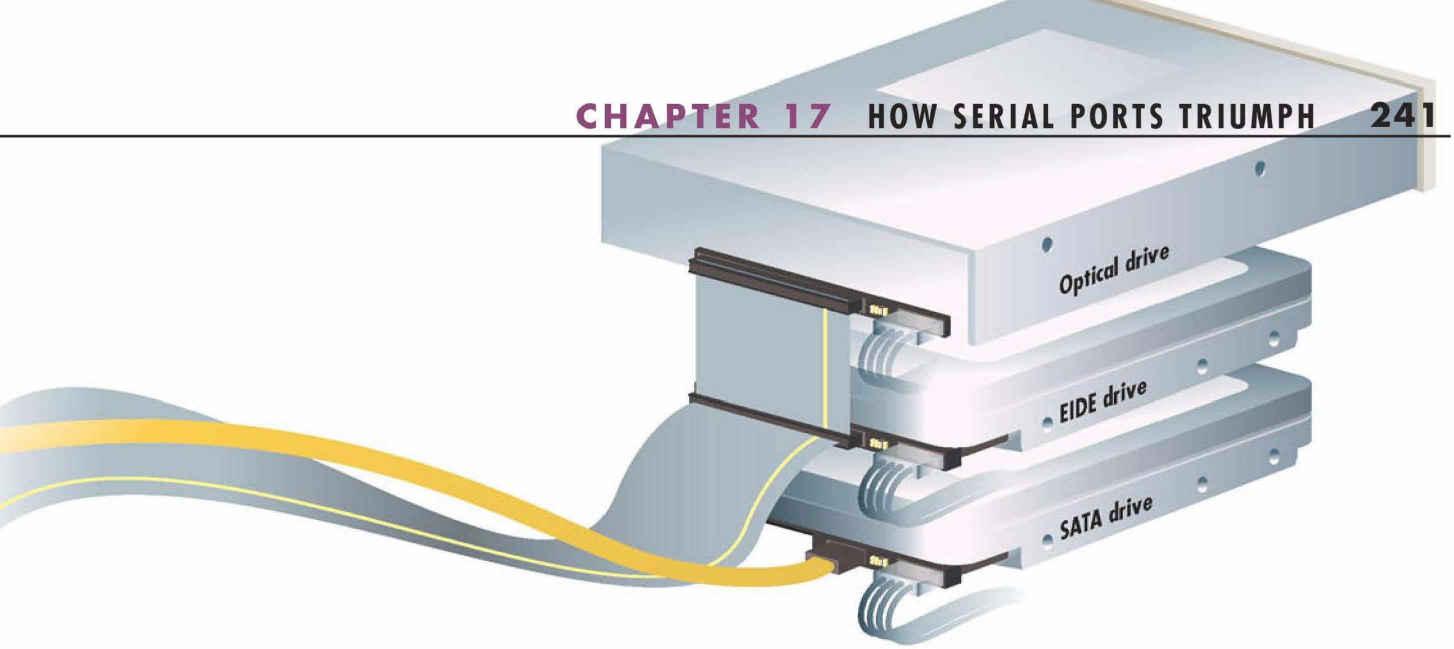
# How SCSI Works



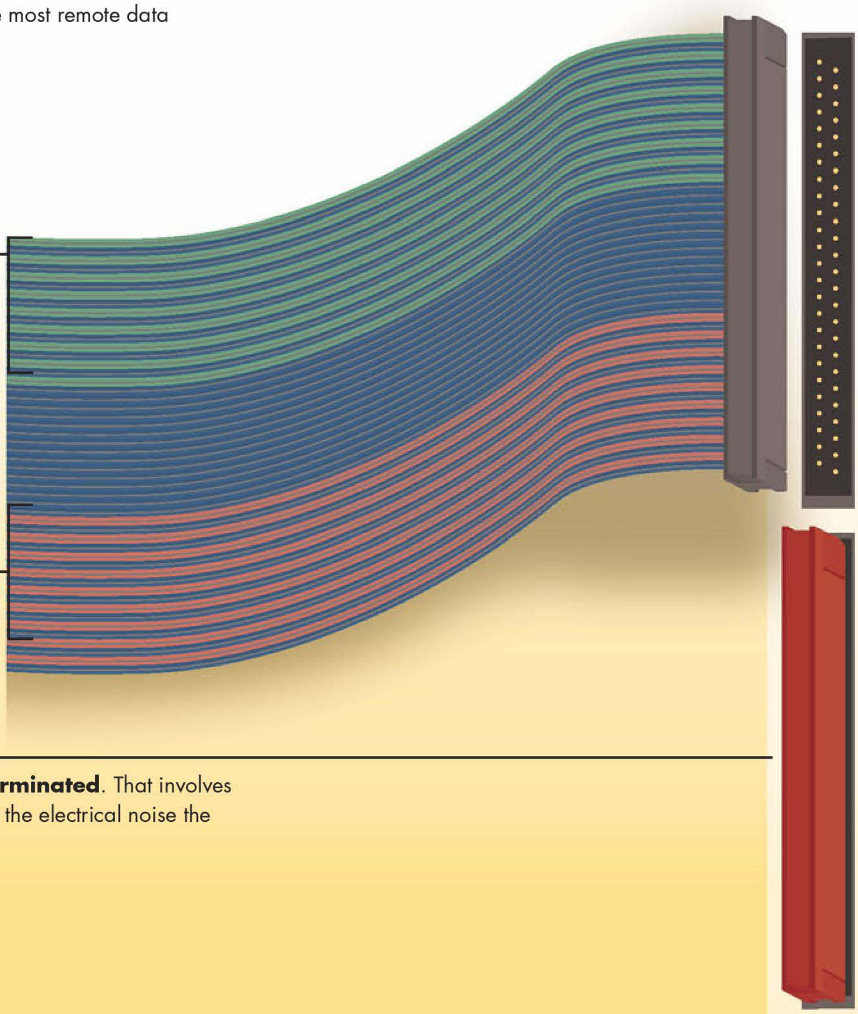

1. Computers such as servers, where there is constant demand for disk files, often use **SCSI (Small Computer System Interface)** because it moves more data in less time than EIDE, and it can pass data among up to 7 to 15 other devices (depending on the type of SCSI). SCSI is also not limited to drives; it connects such equipment as printers and scanners that have enough intelligence to work independently of the CPU.

2. On a SCSI bus, all devices are equal: A printer could initiate a message to the CPU, or a scanner could send a photo to a printer. Any of the devices can communicate with any of the others.






**4** Drives on an SATA connection also improve their performance by using a technique called **Native Command Queing (NCQ)**. This allows the drive to rearrange the order in which it executes up to 32 commands that it receives. The drive can choose to first carry out those commands that call for data nearest the read/write heads and then work out to the most remote data locations.

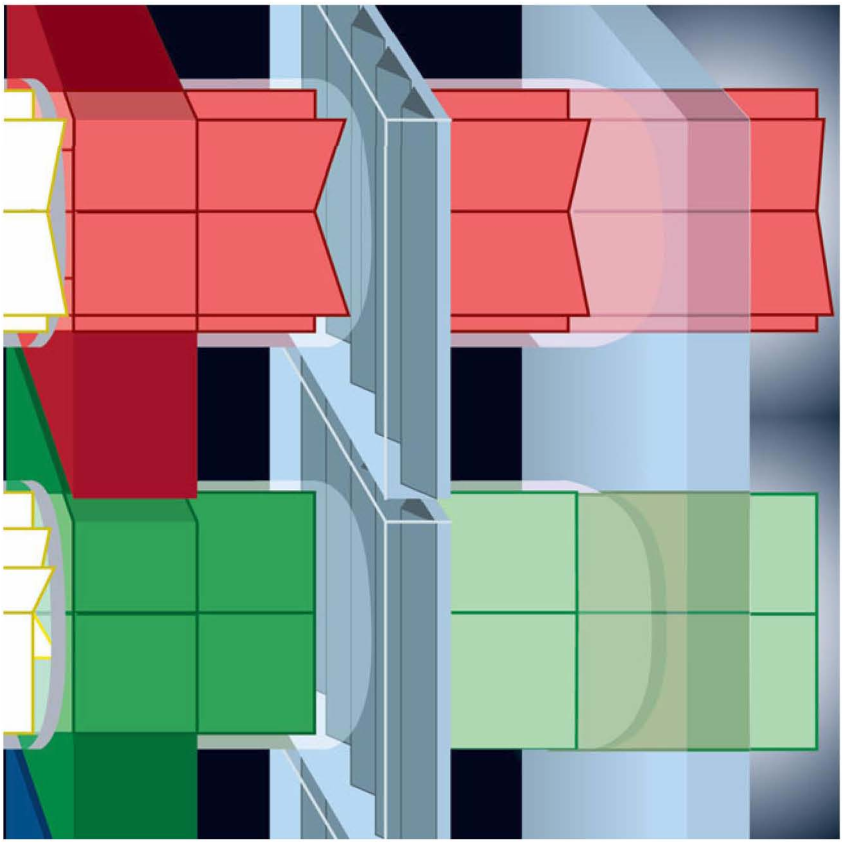


**3** Eight of the wires in an SCSI cable are **data lines**. They carry data in parallel, which is one reason why SCSI is faster then EIDE. **Wide SCSI** uses 16 lines to carry data and makes transfers as fast as 40MB a second. There's also **Ultra2 SCSI** and **Ultra-640 SCSI**, which transfer at rates up to 40MB per second and 640MB per second, respectively.

**4** The devices use another eight wires to send messages among themselves akin to "May I have the floor?" or "This data is going to the printer."

**5** The last SCSI device on the bus must be **terminated**. That involves circuitry or a physical metal cap to **damp** the electrical noise the SCSI cable would generate otherwise.

# CHAPTER 18 How a Computer Display Works



**WHEN** you read the Sunday funnies, you're looking at a hardcopy version of the way a computer displays graphics. Put a magnifying glass to the color comics and you'll see that they are made up of hundreds of dots of red, blue, yellow, and black ink. Different colors and shades are created by varying the sizes of the dots, called **Ben Day dots**. Large red and yellow dots and small blue dots create a shade of orange. Increase the size of the blue dots in the same area, and the color becomes brown. If you look at a comic strip too closely, you see the dots themselves rather than the image they are creating. But hold the comics away from you and the dots resolve themselves into a single image.

A PC monitor works the same way but uses green instead of yellow, and it's additive color as opposed to printed color's subtractive process. Glowing dots of red, green, and blue chemicals blend into millions of colors.

If you were to study a comic strip and make a meticulous record of the position, size, and color of each dot, you would in effect create a non-computerized version of the most common form of computer graphics, a **bitmap**.

As its name implies, a bitmap contains a specific map of all the bits of data—location and color information—that create a computer image by changing the colors in specific pixels on a monitor. (**Pixel** stands for **picture element**, the smallest area of a monitor's screen that can be turned on or off to help create an image.) Bitmaps can be displayed quickly and they're useful when an image is static, as Windows's icons and wallpapers are.

Windows uses graphics and color, largely in the form of bitmaps, to create an interface between you and the operating system. They don't simply make Windows prettier, they convey more information than black-and-white text.

We saw in Chapter 9 how computers store and display bitmaps and vector graphics, which can adapt themselves to size changes and the movement that 3D animation requires. Here, we will look at the two most common ways of displaying output from a computer—the CRT monitor and the liquid crystal display—plus another promising technology: digital light processing.

# How a CRT Paints the Screen

**1** Digital signals from the operating environment or application software go to the **super video graphics array (SVGA)** adapter. The adapter runs the signals through a circuit called a **digital-to-analog converter (DAC)**. Usually, the DAC circuit is contained within one specialized chip that contains three DACs—one for each primary color used in a display: red, blue, and green.

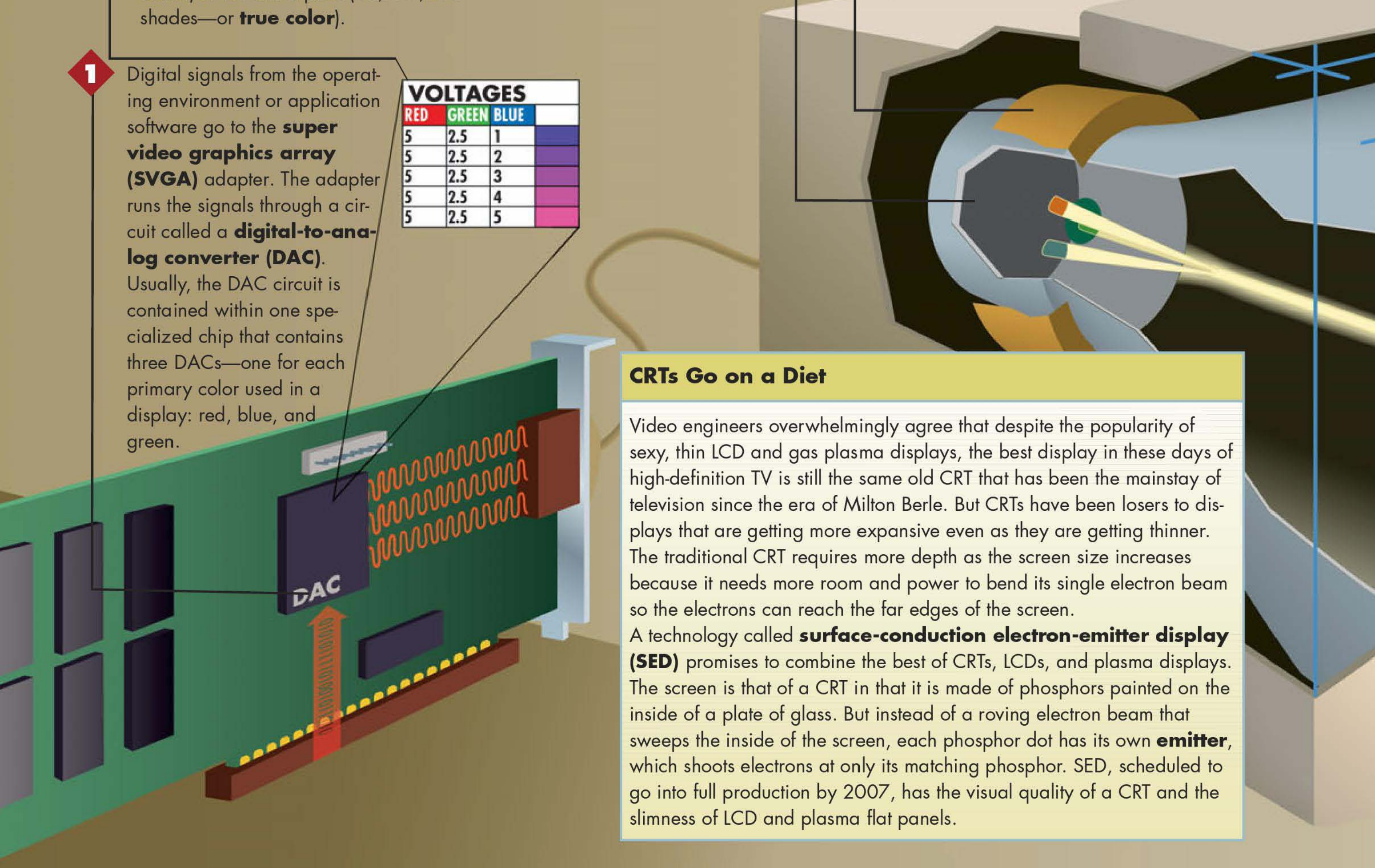
**2** The DAC compares the digital values sent by the PC to a look-up table that contains the matching voltage levels for the three primary colors needed to create the color of a single pixel. In a normal VGA adapter, the table contains values for 262,144 possible colors, of which 256 values can be stored in the VGA adapter's memory at one time. Today's Super-VGA adapters have enough memory to store 16 bits of information for each pixel (65,536 colors, called **high color**) or 24 bits a pixel (16,777,216 shades—or **true color**).

**3** The adapter sends signals to three electron guns located at the back of the monitor's cathode-ray tube (CRT). Through the vacuum inside the CRT, each electron gun shoots out a stream of electrons, one stream for each of the three primary colors. The intensity of each stream is controlled by the signals from the adapter.

**4** The adapter also sends signals to a mechanism in the neck of the CRT that focuses and aims the electron beams. The mechanism, a **magnetic deflection yoke**, uses electromagnetic fields to bend the path of the electron streams. The signals sent to the yoke help determine the monitor's **resolution**—the number of pixels displayed horizontally and vertically—and the monitor's **refresh rate**, which is how frequently the screen's image is redrawn.

VOLTAGES

| RED | GREEN | BLUE | |
|---|---|---|---|
| 5 | 2.5 | 1 | |
| 5 | 2.5 | 2 | |
| 5 | 2.5 | 3 | |
| 5 | 2.5 | 4 | |
| 5 | 2.5 | 5 | |

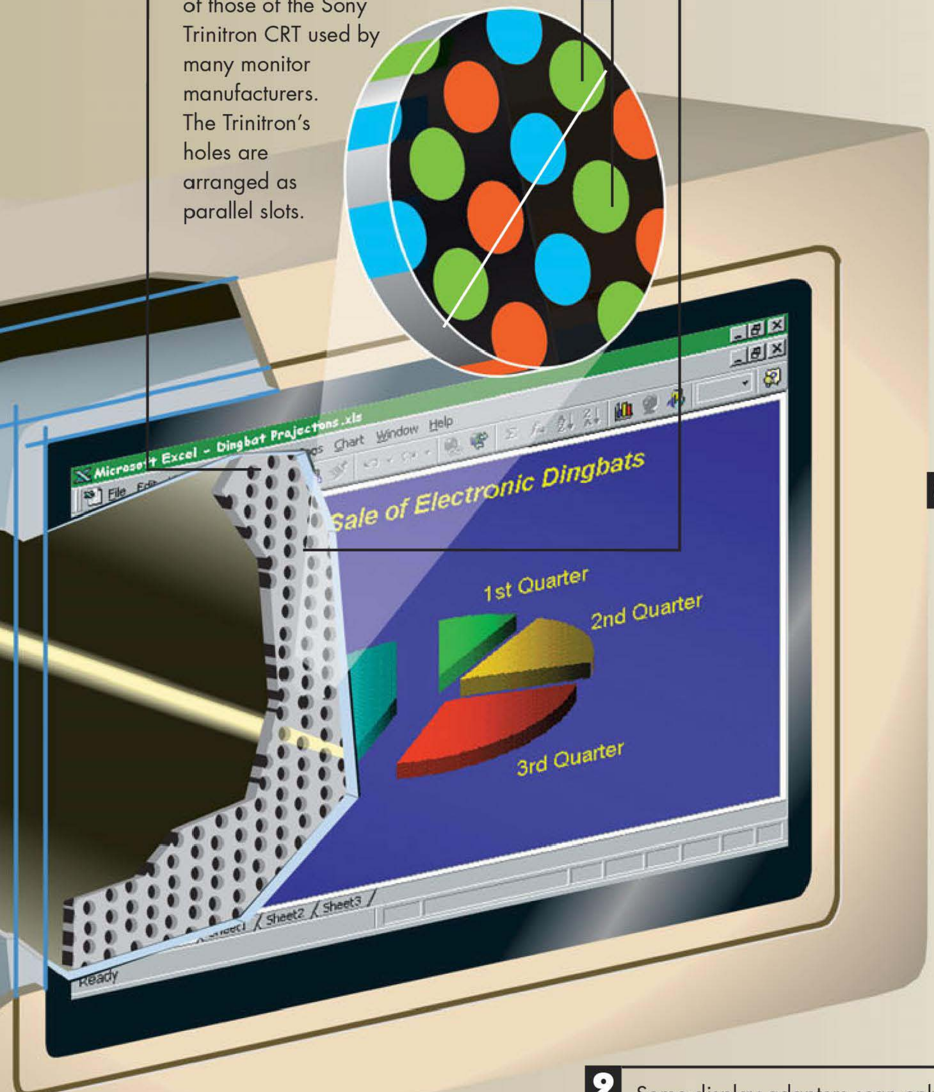



### CRTs Go on a Diet

Video engineers overwhelmingly agree that despite the popularity of sexy, thin LCD and gas plasma displays, the best display in these days of high-definition TV is still the same old CRT that has been the mainstay of television since the era of Milton Berle. But CRTs have been losers to displays that are getting more expansive even as they are getting thinner. The traditional CRT requires more depth as the screen size increases because it needs more room and power to bend its single electron beam so the electrons can reach the far edges of the screen.

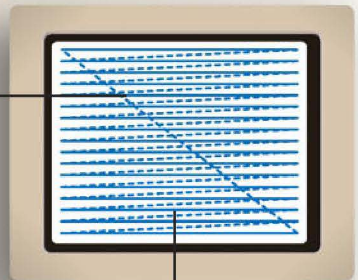
A technology called **surface-conduction electron-emitter display (SED)** promises to combine the best of CRTs, LCDs, and plasma displays. The screen is that of a CRT in that it is made of phosphors painted on the inside of a plate of glass. But instead of a roving electron beam that sweeps the inside of the screen, each phosphor dot has its own **emitter**, which shoots electrons at only its matching phosphor. SED, scheduled to go into full production by 2007, has the visual quality of a CRT and the slimness of LCD and plasma flat panels.

**5** The beams pass through holes in a metal plate called a **shadow mask**. The purpose of the mask is to keep the electron beams precisely aligned with their targets on the inside of the CRT's screen. The CRT's **dot pitch** is the measurement of how close the holes are to each other; the closer the holes, the smaller the dot pitch. This, in turn, creates a sharper image. The holes in most shadow masks are arranged in triangles, with the important exception of those of the Sony Trinitron CRT used by many monitor manufacturers. The Trinitron's holes are arranged as parallel slots.




**6** The electrons strike the phosphors coating the inside of the screen. **Phosphors** are materials that glow when they are struck by electrons. Three different phosphor materials are used—one each for red, blue, and green. The stronger the electron beam that hits a phosphor, the more light the phosphor emits. If each red, green, and blue dot in an arrangement is struck by equally intense electron beams, the result is a dot of white light. To create different colors, the intensity of each of the three beams is varied. After a beam leaves a phosphor dot, the phosphor continues to glow briefly, a condition called **persistence**. For an image to remain stable, the phosphors must be reactivated by repeated scans of the electron beams before the persistence fades away.

**7** After the beams make one horizontal sweep across the screen, the electron streams are turned off as the magnetic yoke refocuses the path of the beams back to the left edge of the screen at a point just below the previous scan line. This process is called **raster scanning**.



**8** The magnetic deflection yoke continually changes the angles at which the electron beams are bent so that they sweep across the entire screen surface from the upper-left corner of the screen to the lower-right corner. A complete sweep of the screen is called a **field**. Upon completing a field, the beams return to the upper-left corner to begin a new field. The screen normally is redrawn, or **refreshed**, about 60 times a second (or higher).

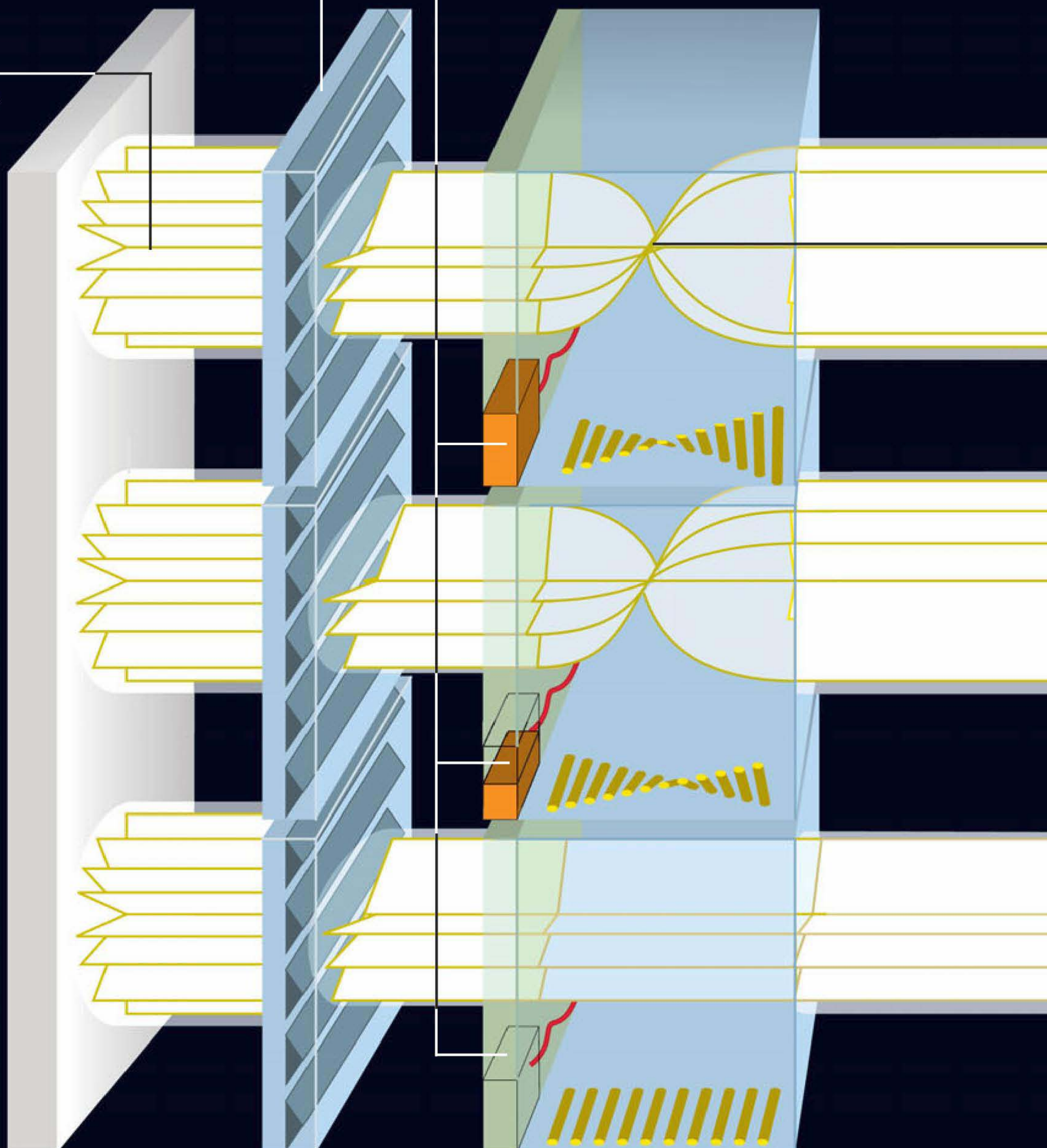
**9** Some display adapters scan only every other line with each field, a process called **interlacing**. Interlacing allows the adapter to create higher resolutions—that is, to scan more lines—with less expensive components. But the fading of the phosphors between each pass can be noticeable, causing the screen to flicker.

# How an LCD Screen Works

**2** A **polarizing filter** in front of the light panel lets through only the light waves that are vibrating more or less horizontally. The fact that the polarizing filter is not entirely precise allows the display to create different hues.

**3** In a layer of liquid-crystal cells—one for each of the three colors that make up a pixel—the graphics adapter applies a varying electrical charge to some of the cells and no charge at all to other cells. In cells to which current is applied, the long, rod-shaped molecules that make up the liquid-crystal material react to the charge by forming a spiral. The greater the charge, the more the molecules twist. With the strongest charge, the molecules at one end of the cell wind up at an angle 90 degrees from the orientation of the molecules at the other end of the cell.

**1** Light emanating from a fluorescent panel behind a portable computer's display panel spreads out in waves that vibrate in all directions.



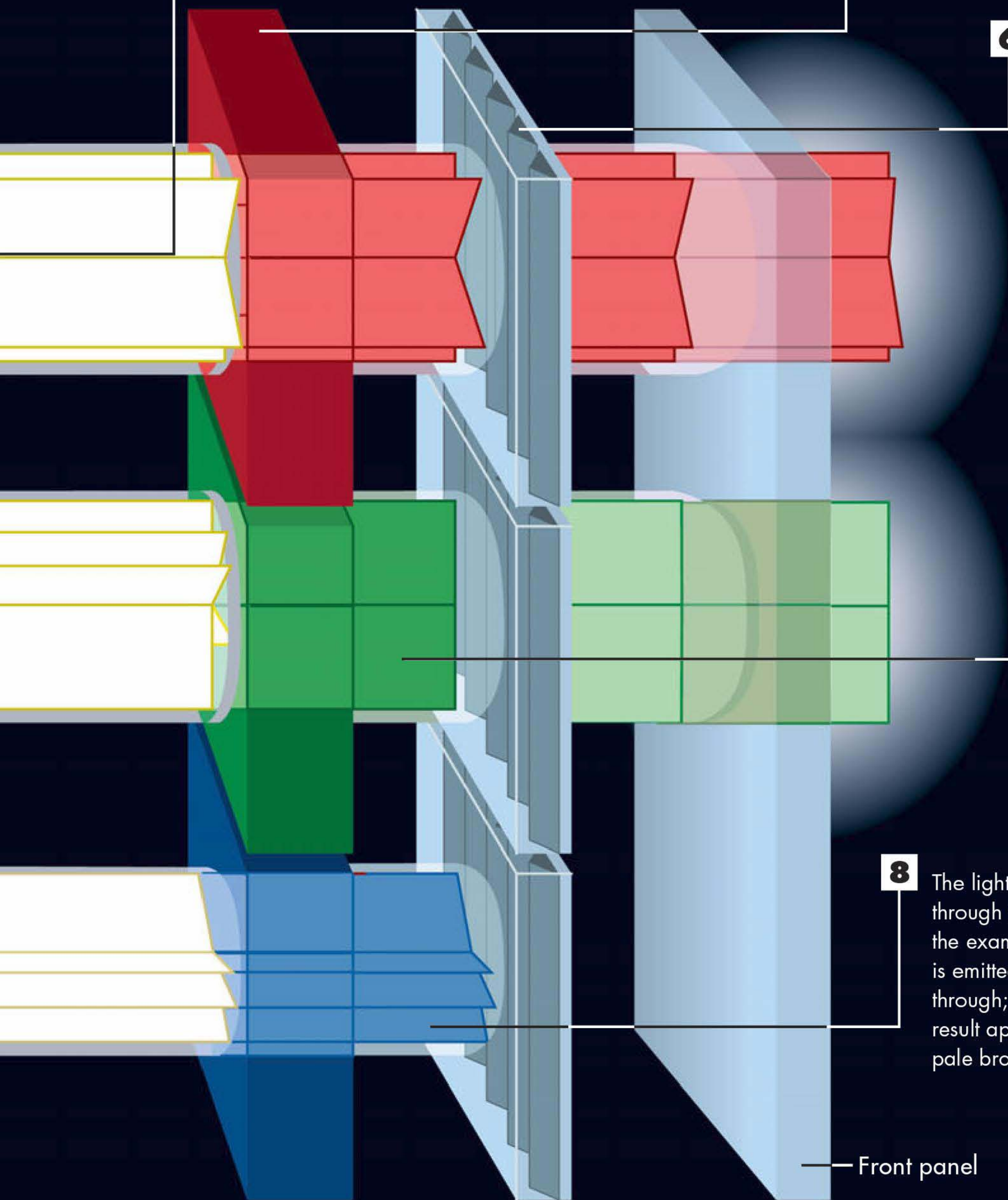
### A Twist on LCD

The model shown here is only one way in which liquid crystals and polarizers can manipulate light. Some LCD panels use two polarizers with the same alignment so that a charge applied to a liquid crystal cell results in light that's blocked because it's twisted. Also, two methods are used to apply charges to liquid crystal cells. **Passive matrix** displays use relatively few electrodes arranged along the edges of the liquid crystal layer and rely on timing to be sure the correct cells are charged. The charges in passive matrix cells fade quickly, causing the colors to look faded. **Active matrix** displays, such as the one shown here, have individual transistors for each cell. The individual transistors provide a more precise and stronger charge, creating more vivid colors.

**4** Polarized light entering the cells from the rear is twisted along the spiral path of the molecules. In the cells to which a full charge was applied, the polarized light emerges, vibrating at a 90-degree angle to its original alignment. Light passing through cells that have no charge emerges unchanged. Cells that received a partial charge twist the light to some angle between 0 and 90 degrees, depending on the amount of the charge.

**5** The light emerging from each liquid-crystal cell passes through one of three color filters—red, blue, or green—that are arranged close to each other.

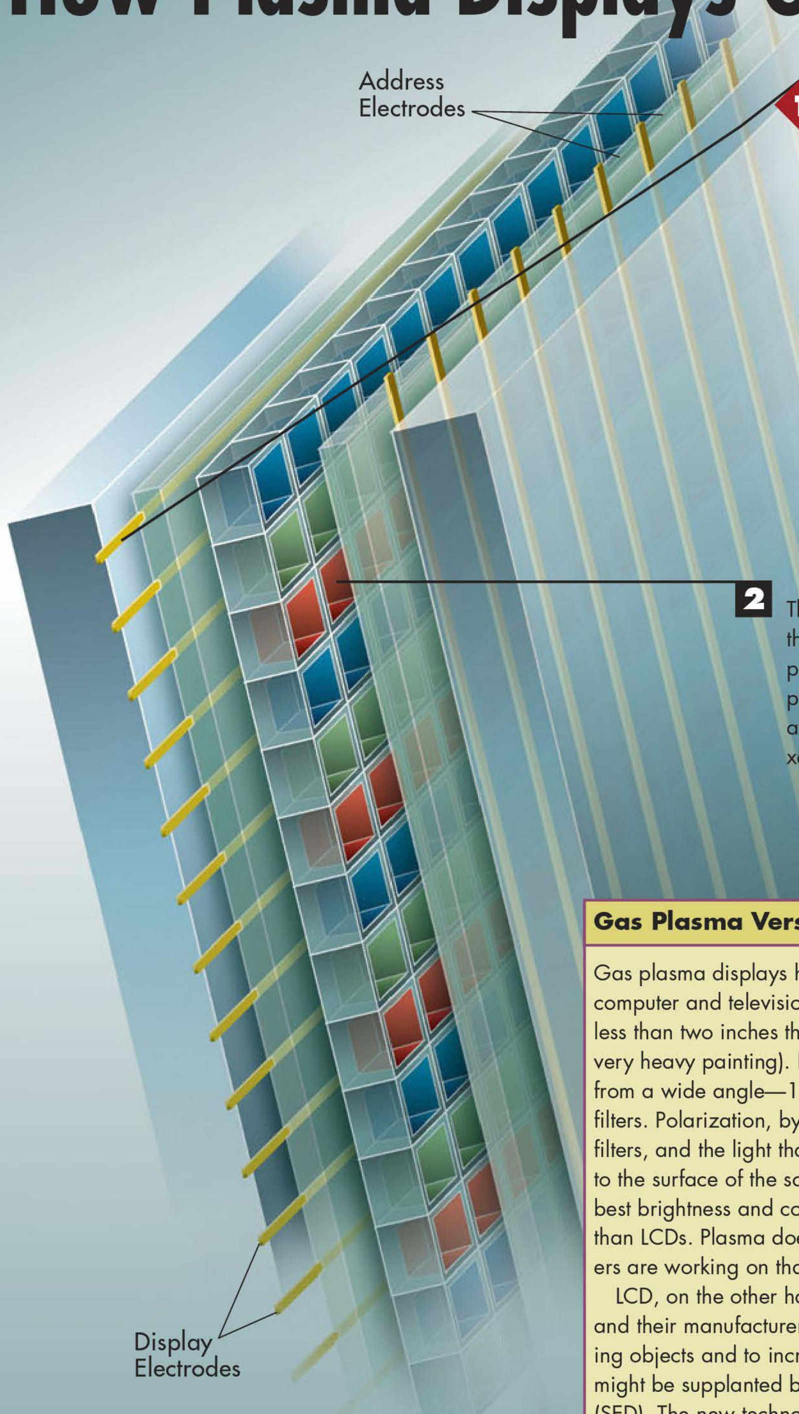
**6** The colored beams of light pass through a second polarizing filter that is aligned to let only light waves that are vibrating more or less vertically pass. The light that passed through a liquid crystal to which a full electrical charge was applied is now oriented perfectly to pass through the second filter.




**7** Because the filter is not entirely precise, some of the light waves that passed through the cell with a partial charge—and which consequently were only partially twisted—pass through the filter while others are blocked.

**8** The light that was not twisted at all when it passed through the liquid crystal is now blocked completely. In the example shown here, 100 percent of the red beam is emitted; 50 percent of the green light makes it through; and the blue light is blocked entirely. The result appears to the human eye as a single point of pale brown light.

# How Plasma Displays Glow




**1** A television receiver or a video card controlling a gas plasma display sends electrical currents to two sets of electrodes. The address electrodes running vertically in the rear of the gas plasma display identify which of the display's pixels are affected when electricity passes through another set of transparent display electrodes, mounted horizontally in the front of the pixels. Running through layers of glass and magnesium oxide, which protect and insulate the electrodes from each other, the electrodes extend the length and width of the rows and columns of pixels that make up the display. The electrodes form intersections that sandwich each of the more than three quarters of a million pixels.

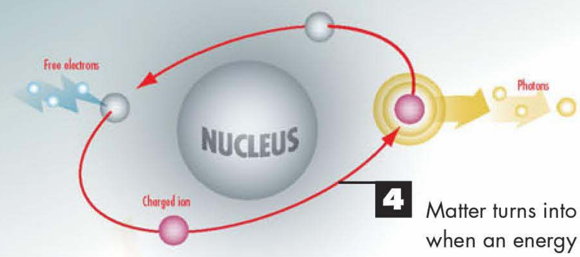
**2** The pixels are cells, depressions in ridges called **ribs** that separate the cells. The walls of the cells are spray-painted on the inside with one of three types of phosphors that glow either red, blue, or green when they are energized. Trapped inside each cell is a mixture of xenon and neon gases.

**Gas Plasma Versus LCD Versus...**

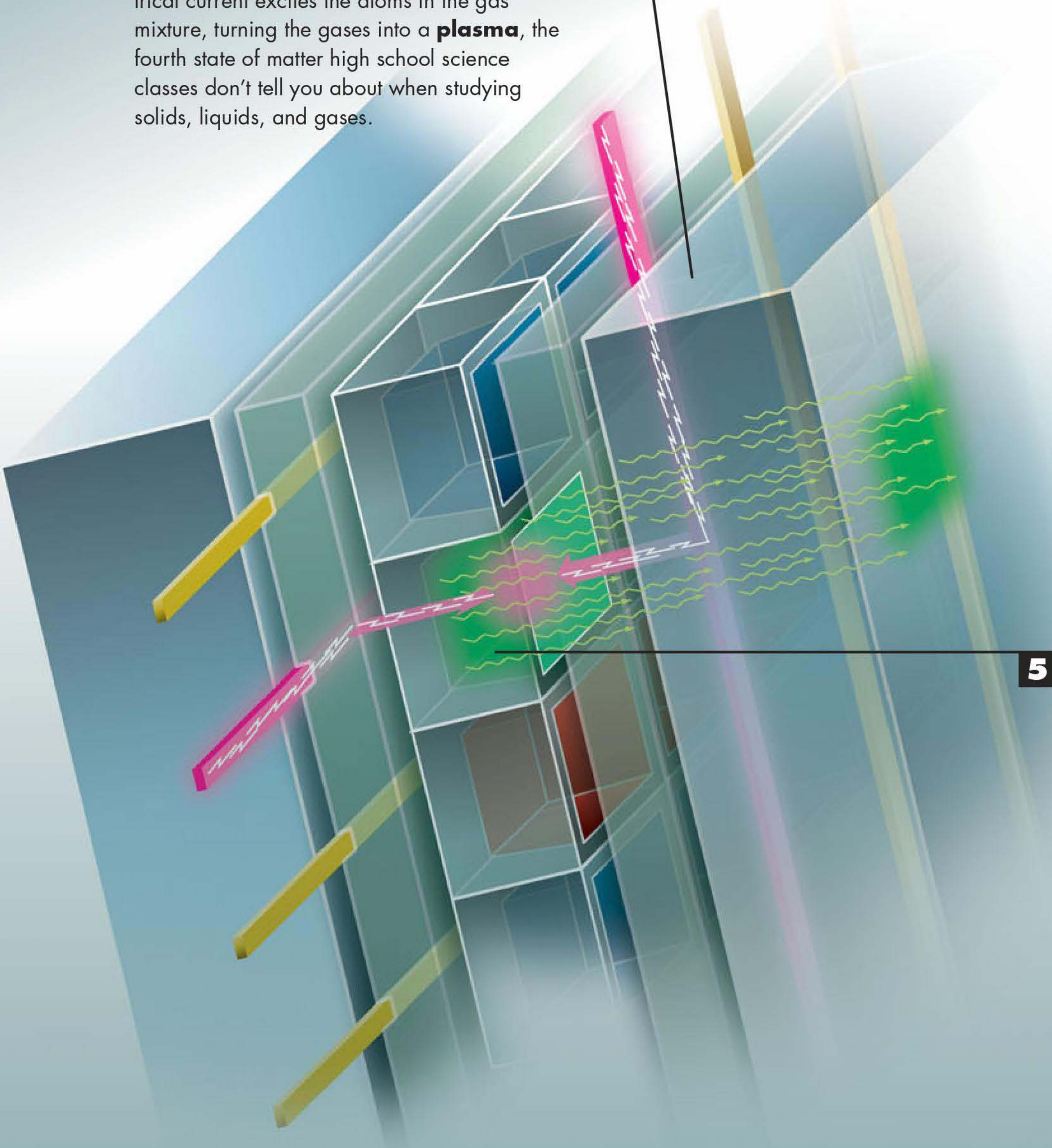
Gas plasma displays have a lot going for them in the race for flat, thin computer and television displays. Like an LCD screen, a plasma display is less than two inches thick. You can hang it on the wall like a painting (a very heavy painting). But unlike LCD, plasma is brighter and can be seen from a wide angle—160°—because light does not pass through polarizing filters. Polarization, by its nature, absorbs most of the light that strikes the filters, and the light that does pass through travels mostly at a right angle to the surface of the screen, requiring you to view the screen straight-on for best brightness and contrast. Gas plasma displays can easily be larger than LCDs. Plasma does, however, have a problem with burn-in, but makers are working on that.

LCD, on the other hand, has price advantages, creates lighter displays, and their manufacturers are working to overcome the blurring of fast-moving objects and to increase the view angle. But both LCD and plasma might be supplanted by surface-conduction electron-emission displays (SED). The new technology is essentially a flat CRT screen with a separate electron gun for each pixel. See the illustration on CRT, "How a CRT Paints the Screen," p. 244.

**3** When the display controller wants a particular pixel to glow, it **opens** the **address line** that leads to that pixel's cell. (Opening is accomplished by **closing** a circuit so electricity can flow through it.) At the same time, the controller sends a stream of electricity down the **display line** leading to the same pixel. The electricity, attracted by the charge on the open address line, jumps through the cell to complete a circuit with it. The energy from the electrical current excites the atoms in the gas mixture, turning the gases into a **plasma**, the fourth state of matter high school science classes don't tell you about when studying solids, liquids, and gases.

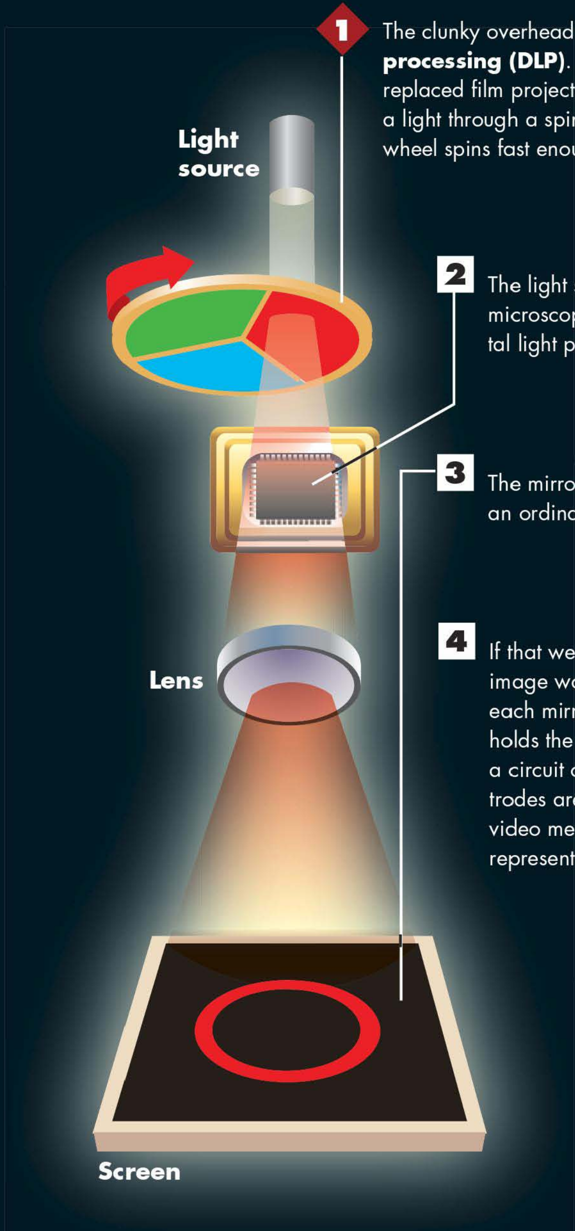



**4** Matter turns into a plasma when an energy source excites normally stable gases, such as neon and xenon. Free-running electrons from the electricity strike the gas atoms, creating and imparting their energy to ions, which are atoms that have become positively or negatively charged because of an imbalance of electrons. The ions are unstable and revert to their normal state. When they do, they emit the energy that created them in the form of ultraviolet bullets of light called **photons**.



**5** The ultraviolet photons strike the phosphors on the walls of the cell. The photon's energy excites the phosphors so they glow—similar to the process in an everyday cathode-ray tube monitor. Different phosphor materials that glow either red, blue, or green coat the inside of adjacent cells to create a single logical pixel. By varying the amount of current going to each cell, the display changes the mix of the three primary colors to create different hues.

# How Digital Light Processing Works




**1** The clunky overhead transparency projector is being replaced by **digital light processing (DLP)**. DLP projects brighter, sharper images and have even replaced film projectors in some specialized movie theaters. It works by shining a light through a spinning wheel divided into red, blue, and green filters. The wheel spins fast enough so that each color is produced 60 times a second.

**2** The light strikes a panel of up to 1.3 million microscopic mirrors on the surface of a digital light processing (DLP) chip.

**3** The mirrors reflect the light through a lens to an ordinary projection screen.

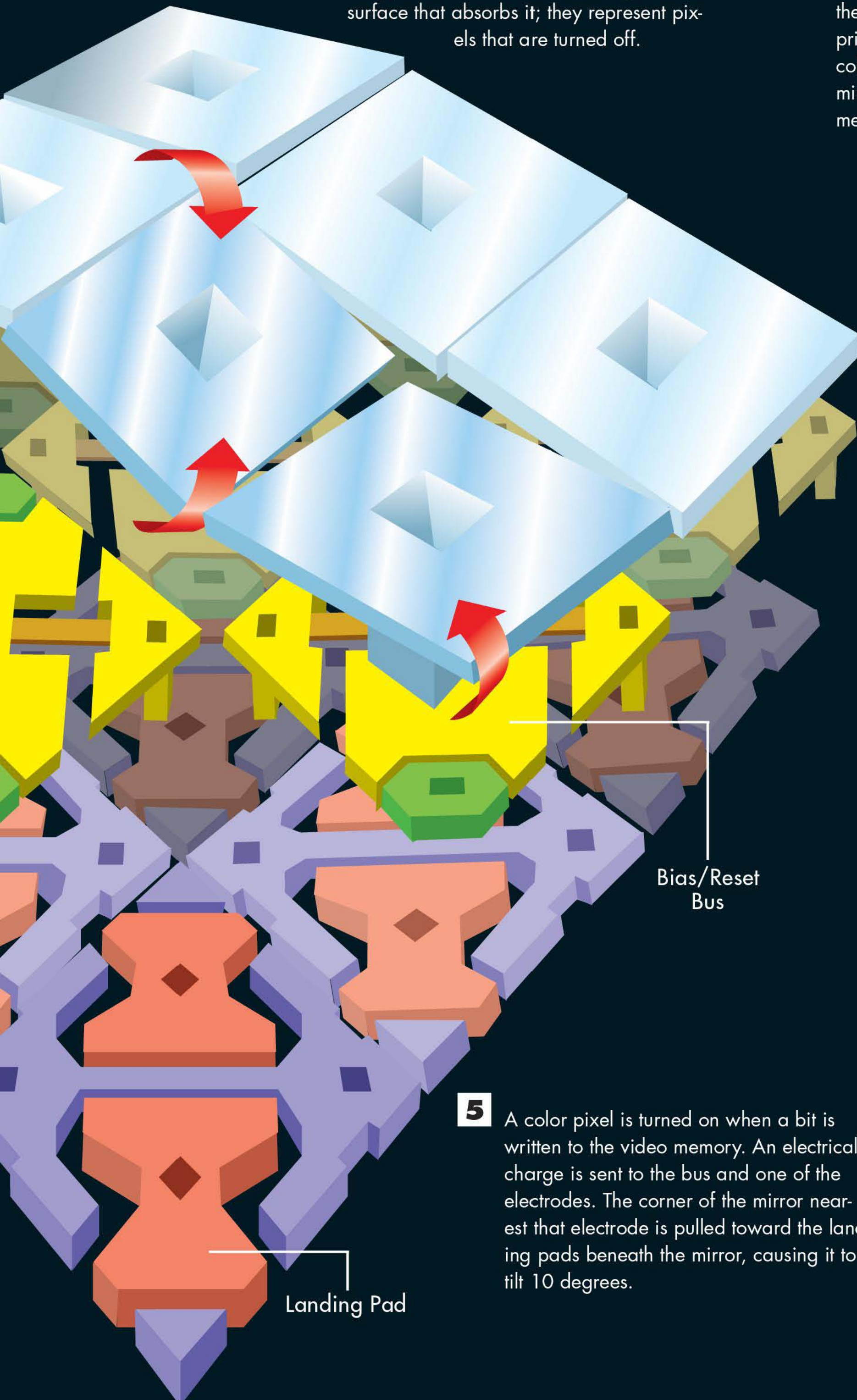
**4** If that were all the mirrors did, the projected image would be a large white rectangle. But each mirror is attached to a flexible hinge that holds the mirror above a pair of electrodes and a circuit called the **bias/reset bus**. The electrodes are connected directly to an array of video memory bits so that each electrode can represent a 1 or a 0.

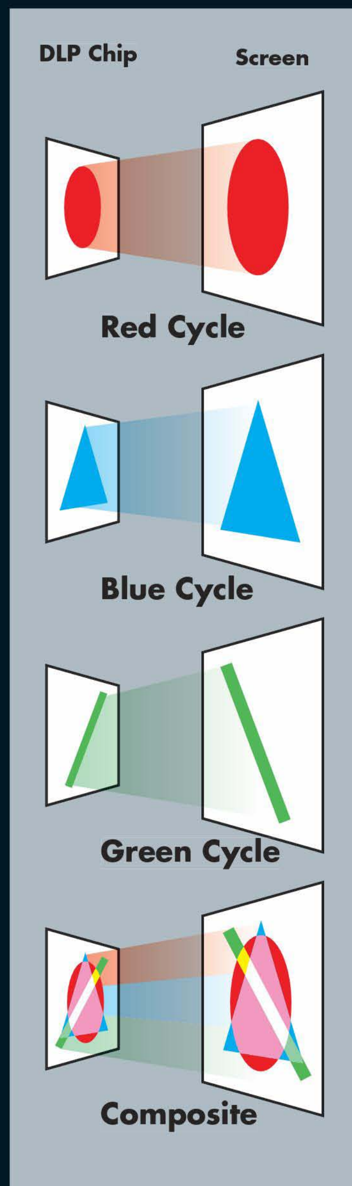
### LCD Projector

Another way of casting a computer display image is with a liquid-crystal display projector. Colors and images are created the same way for an LCD project as they are for a laptop computer's LCD panel. The essential differences are that the projector's light source is much stronger than a laptop's, and the image is focused through a lens. But the nature of an LCD means much of the light is absorbed so that an LCD image is dimmer than a DLP projection. In addition, the gaps between the mirrors in a DLP display are smaller than those between the cells that form the LCD, making the DLP image sharper.

**6** Mirrors, functioning as pixels that are turned on, tilt one way to reflect light toward a screen. Mirrors tilted in the opposite direction reflect the light to a surface that absorbs it; they represent pixels that are turned off.

**7** Hues of colors are created by how often the source light is turned on and off while the three filters pass in front of it. The color wheel and the RGB data are synched so that the amount of time a mirror reflects any one of the three primary colors is in proportion to the amount of time that color in the mixture is perceived by the human eye. Each mirror can tilt on or off in less than 20 milliseconds, which means that each mirror can generate 16.7 million colors.




**5** A color pixel is turned on when a bit is written to the video memory. An electrical charge is sent to the bus and one of the electrodes. The corner of the mirror nearest that electrode is pulled toward the landing pads beneath the mirror, causing it to tilt 10 degrees.

# CHAPTER 19 How Digital Photography Works

**PUT** a digital camera next to a comparably priced camera that still uses film. What's the difference? Certainly it's not anything you can see: They both have a lens, some sort of viewfinder to peer through, and similar assortments of buttons and knobs.

The important difference between the two cameras is buried inside them. Take off the back of the film camera and you'll see a slot at one end where you insert your cassette of film and an empty spool on the other end to take up the roll of film as each frame is exposed. Between them is the shutter. When the camera back is closed, the film is held firmly against the frame around the shutter by a smooth flat surface called the *pressure plate*.

Take a digital camera and open the back—and you can't! There is no way to open it and see what's inside. That's what this chapter is for, to show you what you ordinarily can't see on your own. What you'll see in this chapter is that all the apparatus inside a film camera—and the film itself—have been replaced by a **microchip** packed with microscopic electronic switches called **transistors**. This chip is similar to the computer microchips you've been reading about in this book.

This particular type of microchip, as you'll see in more detail later in the book, is covered with a special type of transistor—millions of transistors, actually—that is sensitive to light and converts that light into electricity. The chip is called an **image sensor**, or an **imager**.

The image sensor's ability to translate different colors and intensities of light into constantly changing electrical currents is what accounts for the other important differences between digital and film cameras. The most obvious is that most digital cameras have an LCD screen on the back of them, like a tiny TV set, where the camera displays the scene to be shot or the photographs already stored in the camera. The LCD has its own array of transistors that do just the opposite of the imager's transistors: They convert electricity into light. (More about that, too, later on.)

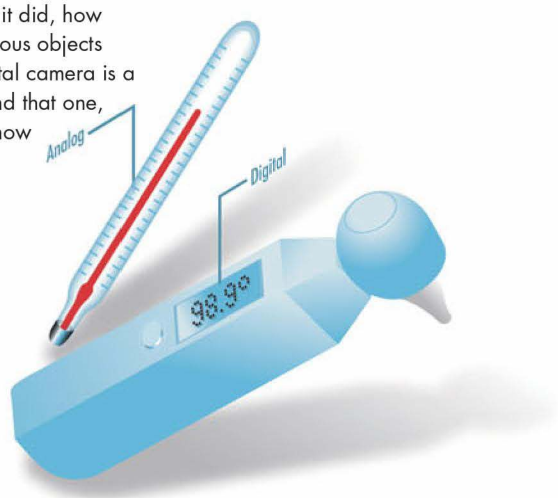
If you inspect the two cameras closely enough, you might find some other differences. The digital camera might have, for example, a button or switch for something called **white balance**. It might have controls for onscreen menus, or for displaying information about a shot you've snapped, or a button with an icon of a trash can that's used for deleting files.

But that's about it. Those are all the differences you'll find by visually inspecting your digital camera—even if you tear it apart. With few exceptions, such as the aforementioned white balance, you use a digital camera just as you would a film camera.

But after you click the shutter, letting light fleetingly strike the image sensor, you've created not just one picture, but the possibility for scores of pictures. The future of that picture is limited only by your imagination.

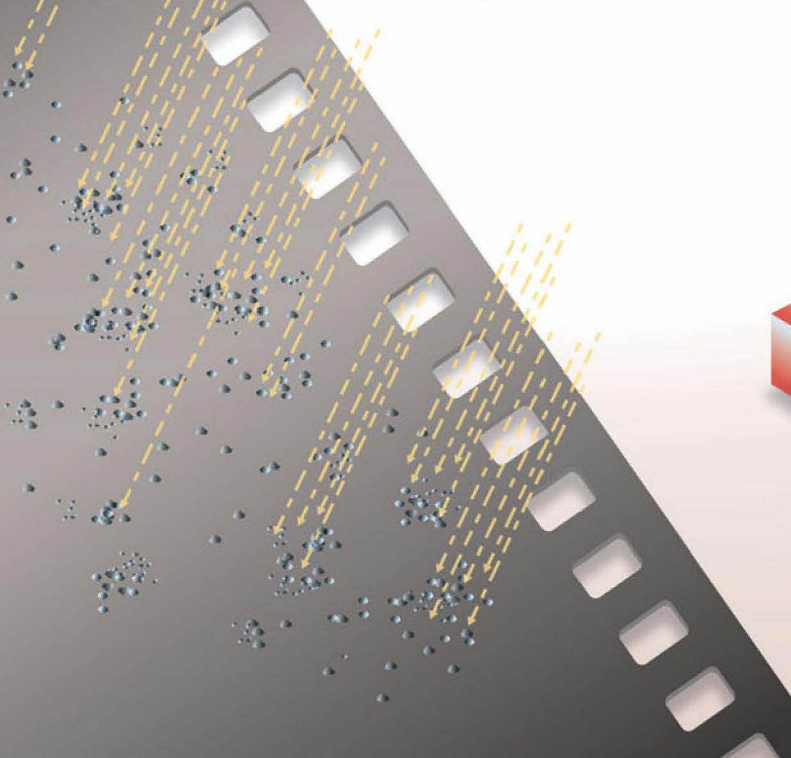
# How Digital Cameras Capture Light

The street you're driving down doesn't suddenly end here and start again over *there*, with no way to bridge the gap. Time doesn't stop for 5 minutes and then pick up again where it left off. Of course, if it did, how would we know? The point is that we're used to thinking of things as analog—smooth, continuous objects without any quantum gaps between here and there. But in the computer world—and your digital camera is a computer—nothing is smooth and continuous. It's digital. There are gaps between this point and that one, between this moment and the next. Before we can do all the wonderful things available to us now that a computer is packed into our camera, we and our cameras have to communicate— we with our words; the cameras with a mathematical alphabet of only 0s and 1s.

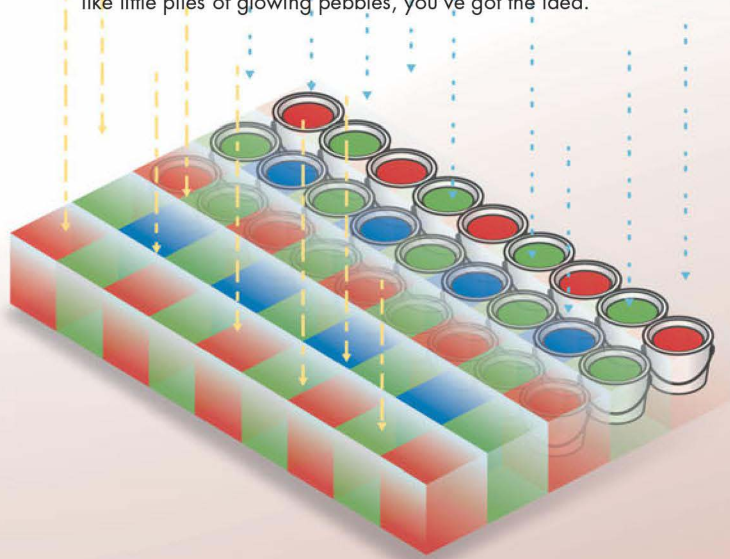



**1** Anything in the universe can be measured in analog or digital terms. **Analog** simply means that the expression of the measurement is analogous to whatever it's measuring. An old-fashioned thermometer with red-dyed alcohol in a tube gives an analog representation of how hot it is because the alcohol rises, literally, as the temperature rises. A **digital** thermometer expresses the temperature in numbers on a small LCD screen. The numbers themselves don't grow larger as the temperature rises.

**2** Film is an analog method of recording a photo. Where light strikes the silver halide crystals embedded in film, the crystals clump together. Where the light is stronger, more crystals clump. Where the light is dimmer, fewer crystals clump.

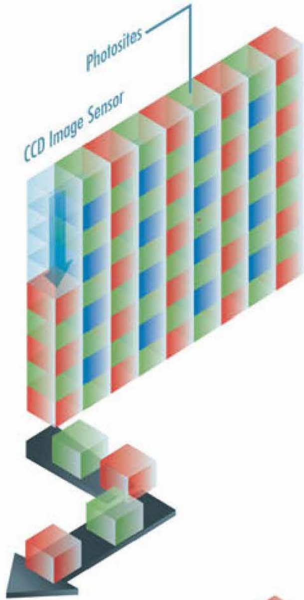


**3** The photodiodes that replace film in a digital camera don't look any different after a picture is snapped than they did before. They don't shift about on the surface of the image sensor to clump where the light is stronger. But there is an unseen analog process afoot when a digital photo is taken. Each of the photodiodes collects photons of light as long as the shutter is open. The brighter a part of a photograph is, the more photons hit the pixels that are analogous to that part of the scene. When the shutter closes, all the pixels have electrical charges that are proportional to the amount of light they received. If you picture the photons piling up like little piles of glowing pebbles, you've got the idea.

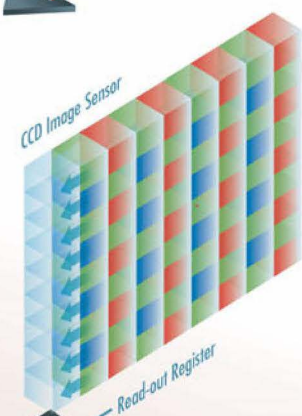


What happens next depends on whether the image sensor is a **CCD (charged coupled device)** or **CMOS (complementary metal oxide semiconductor)**. Don't bother with the full-fledged names. Everyone uses the acronyms, and they won't be on the quiz. You will hear a lot of techno-hype from camera makers citing reasons one technology is better than the other. You can ignore that, too. The type of imager is just one factor that contributes to the photo that eventually will come out of your printer. It's the print that's important. Whether you're happy with it or not doesn't hinge on the type of image sensor. But here, for the sheer joy of knowledge alone, are the differences in how the two types of chips work.

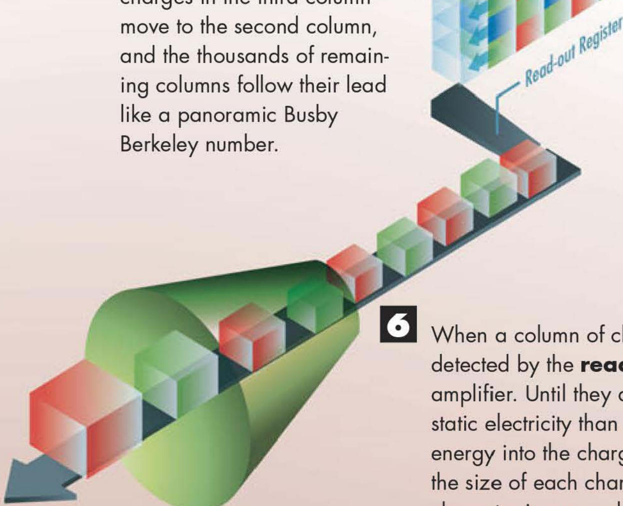
## CCD




**4** The charges in an **interline CCD** imager, which is what most CCDs are, begin an orderly procession toward their future existence as digital numbers like well-behaved schoolchildren in a fire drill. At one end of the imager, the charges move down and out at the bottom of the column as if someone were continually pulling the bottom can of Coke out of a dispenser.

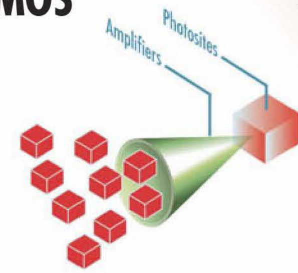



**5** When the last charge has rolled out of the bottom of the column, the charges in the second column shift to fill the vacancies left by the newly departed charges. The charges in the third column move to the second column, and the thousands of remaining columns follow their lead like a panoramic Busby Berkeley number.

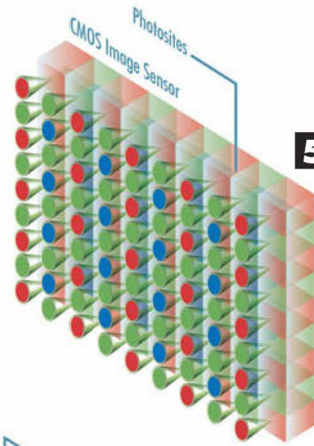



**6** When a column of charges falls out of the imager, it is detected by the **read-out register**, which takes them to an amplifier. Until they are amplified, the charges are more faint static electricity than electrical current. The amplifier pumps energy into the charges giving them a voltage in proportion to the size of each charge, much as a flagging video game character is pumped up with "life force" by jumping on a coin.

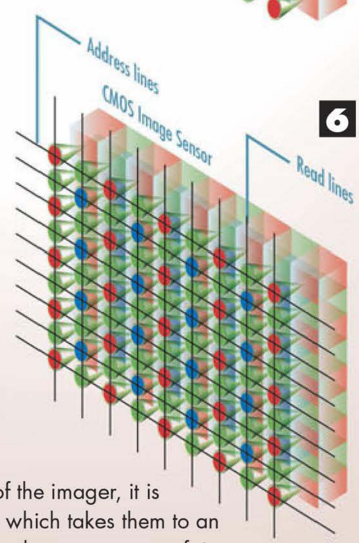
## CMOS




**4** Unlike the photosites in CCDs—pretty much passive **capacitors** that do little but store an electrical charge until a control somewhere else tells them what to do with it—a CMOS sensor is able on its own to do some of the processing necessary to make something useful out of the charges the photosites have obtained from the light.

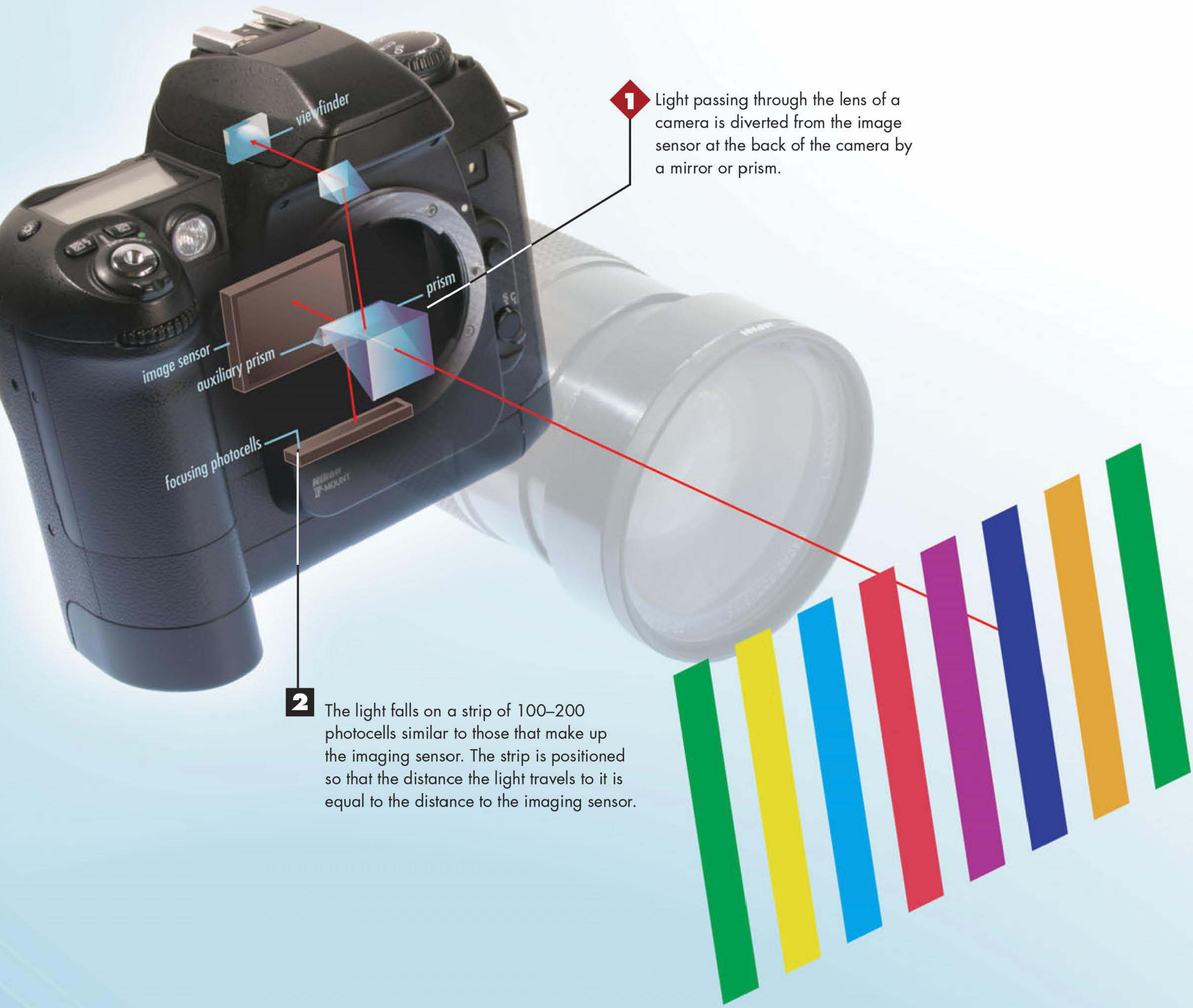



**5** The first thing the CMOS image sensor does is use the amplifiers that are part of each photosite. This eliminates the need for the charges to go through an amplifier in single file after they've left the sensor.



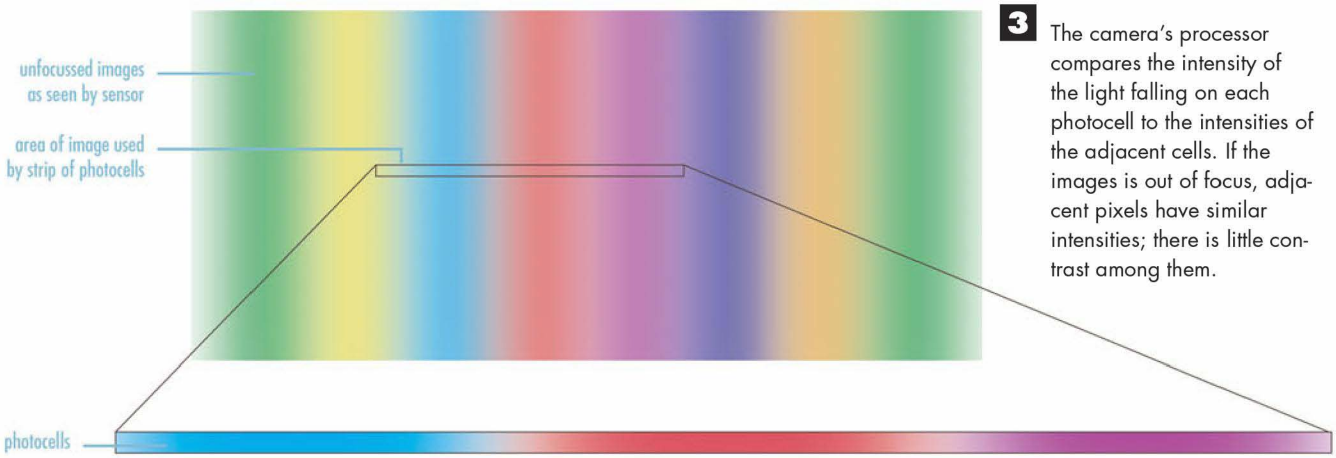

**6** More importantly, the onsite amplifier eliminates the slow classroom drill CCDs use to leave their nest. As soon as the amplifiers have turned the charges into actual voltages, those voltages are read over a grid of X-Y wires whose intersections correspond to locations of the photosites. It's the voltages' way of saying simultaneously, "Beam us up."

# How Autofocus Lenses Work



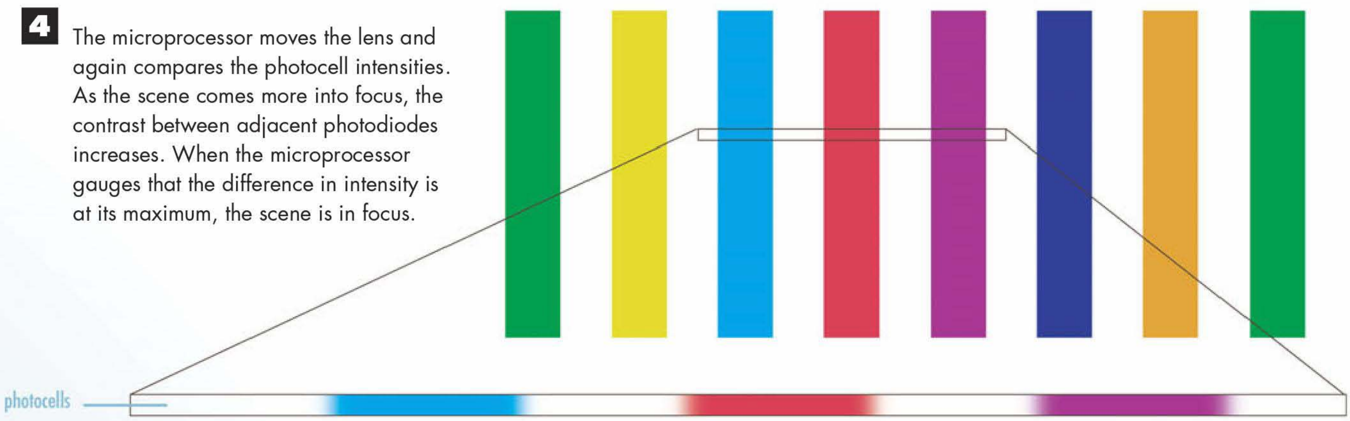

**1** Light passing through the lens of a camera is diverted from the image sensor at the back of the camera by a mirror or prism.

**2** The light falls on a strip of 100–200 photocells similar to those that make up the imaging sensor. The strip is positioned so that the distance the light travels to it is equal to the distance to the imaging sensor.




**3** The camera's processor compares the intensity of the light falling on each photocell to the intensities of the adjacent cells. If the images is out of focus, adjacent pixels have similar intensities; there is little contrast among them.

**4** The microprocessor moves the lens and again compares the photocell intensities. As the scene comes more into focus, the contrast between adjacent photodiodes increases. When the microprocessor gauges that the difference in intensity is at its maximum, the scene is in focus.




### Autofocus Limitations

Both passive and active autofocus have advantages and disadvantages. Active focusing works at night and in dim lighting, but the infrared light can bounce off glass or mirrors, confusing the camera's processor. Using passive focusing, you aim through windows and there are no distance limitations beyond which it cannot work. But a blank wall or a scene devoid of straight edges, particularly vertical lines, throws passive autofocus for a loop.

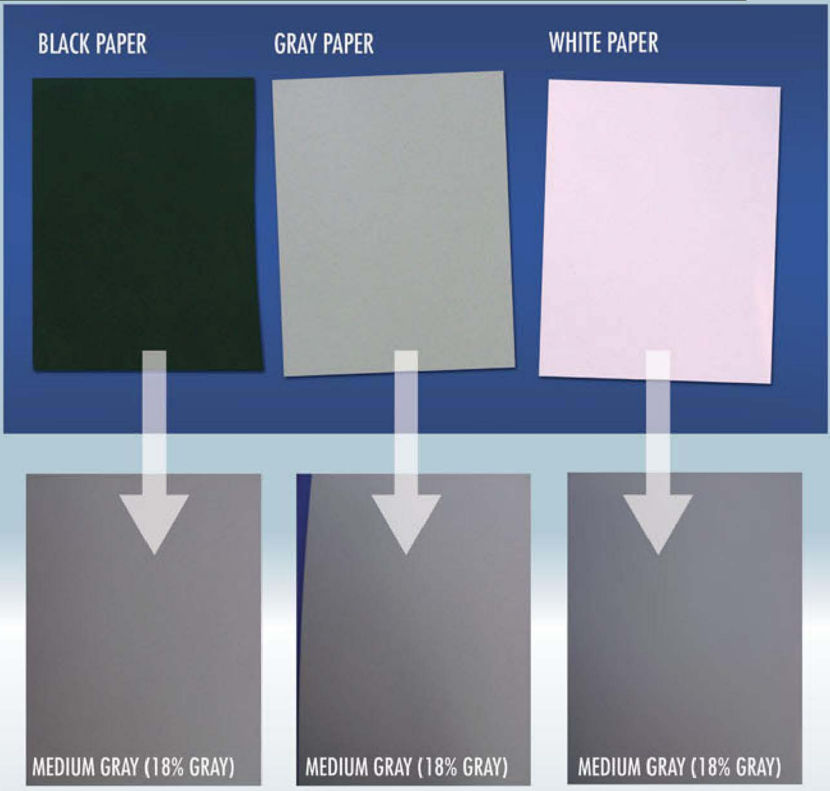
To minimize the effects of shooting through glass, the photographer can put the lens directly on the glass. The infrared light passes through the glass. Any light that bounces back makes the trip too quickly for the camera to use its timing information. With passive autofocus, turning the camera 90° can give the camera the perpendicular lines it needs. In scenes with little contrast, try focusing on an object elsewhere about the same distance away as your subject. Then keep the shutter button pressed down about halfway as you turn to frame your real subject. On some cameras, holding the button locks the focus until you press the button all the way to shoot your photo or until you release it. The camera's processor compares the intensity of the light falling on each photocell to the intensities of the adjacent cells. If the image is out of focus, adjacent pixels have similar intensities; there is little contrast among them.

# How Auto Exposure Works

The most complex part of a digital camera is its exposure system. It's more than a photodiode measuring the light coming through the lens. Among the most versatile cameras, the exposure system has more than one way to measure light and mixes those measurements into a brew made of settings for the type of lighting; the sensitivity of the image sensor; and special settings for action shots, fireworks, black-and-white, or even special effects such as sepia toning. That brew is siphoned to set into action the diaphragm and shutter, all in the blink of an eye.

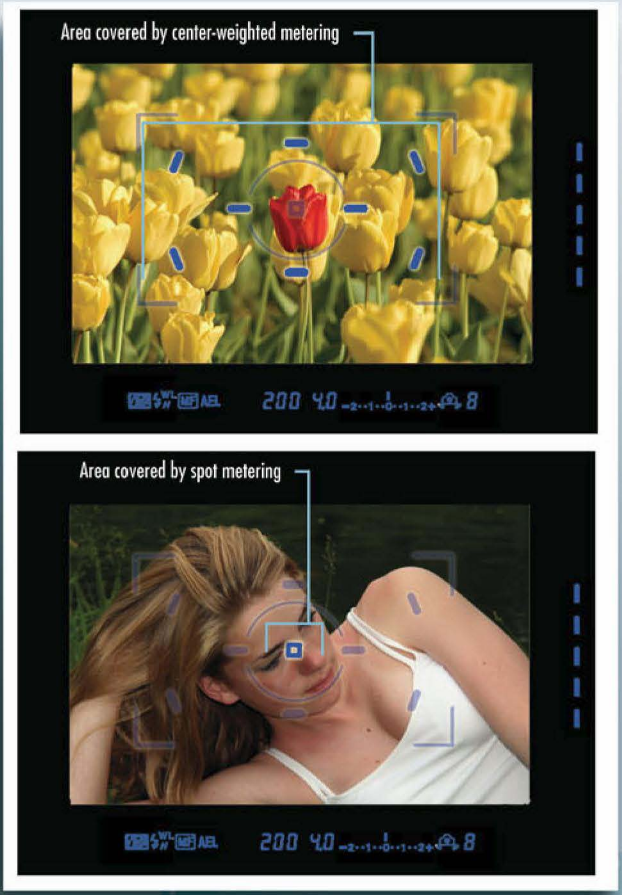
1 Less-expensive digital cameras—called **point-and-shoot (POS)** models—often have only one of the many ways of measuring light that more-expensive cameras boast. It's called **full-frame** and it typically uses one photodiode device mounted next to the lens. On better cameras, full-frame metering uses several photodiodes—as many as 45—mounted in the path of the light on its way from the lens to the shutter hiding the image sensor. Either type of full-frame averages the intensities of light reflected off a subject to determine a shutter speed and aperture that will produce an exposure that is expected to render everything in the photo. But unless a scene is evenly lit and contains only subjects with the same color value, full-frame exposures are usually less accurate.


2 In the photo here, for example, the bright sunlight falling on the bricks behind the boy riding in his car has made the camera's auto exposure feature overcompensate and shut down the diaphragm too much. The result is muddy shadows revealing little detail in the most important part of the photo.




**3** This occurs because exposure systems think that no matter how bright or how dim something is, it is 18% gray (which is considered a medium gray). In these photographs, you can see white, gray, and black sheets of paper in their true colors when they are all in the same photo. But when each is photographed so that it is the only object measured by the exposure meter, the camera's exposure automatically is set to render the three sheets of paper as the same medium gray.

**4** To overcome the perils of using an average of an entire scene's illumination, better digital cameras have alternative ways of measuring light that can be chosen from a menu displayed in the camera's LCD screen or by one of the camera's control knobs or buttons. The first alternative is **center-weighted** measurement. It meters the light in an area that amounts to about a tenth of the total photo area. As the name implies, that tenth is located in the center of the screen on the theory that that's where the most important part of the scene is.




**5** The other common alternative is **spot metering**, which gives the photographer greater ability to expose the part of the scene that is most crucial. Illumination is read from only a small circle in the center of the screen, allowing the photographer to expose for the lighting on a cheek that might be surrounded by dark hair and a beard that would otherwise overwhelm the light readings. When the crucial subject matter is off-center, some cameras have the capability to make small areas on different parts of the image the spot for purposes of metering. For those without such cameras, you can use the **shutter lock** described in the box below.

### The Half-push

When the most important element of a photo is not in the center of the frame, most cameras have a shutter lock feature that lets them focus and get an exposure reading for their picture by aiming the center at that important element. Then, by pushing the shutter button only halfway, the autofocus and auto exposure settings are locked. The photographer then reframes the picture with that key element away from the center and presses the shutter button the rest of the way.

**45,000 B.C.** Neanderthal man carves on woolly mammoth tooth, discovered near Tata, Hungary.

**28,000 B.C.** Cro-Magnon notation, possibly of phases of the moon, carved onto bone, discovered at Blanchard, France.

**3400-3100 B.C.** Inscription on Mesopotamian tokens overlap with pictography.

**10,000 B.C.** Antler baton engraved with seal, salmon, and plants portrayed, discovered at Montgaudier, France.

**1270 B.C.** Syrian scholar compiles an encyclopedia.

**950** Bored women in a Chinese harem invent playing cards.

**1560** In Italy, the portable camera obscura allows precise tracing of an image.

**1794** Panorama, forerunner of movie theaters, opens.

**1807** Camera lucida improves image tracing.

**1872-1877** Eadweard Muybridge shoots a series of motion photographs, which can be viewed by mounting them to a stroboscopic disc.

**1877** Thomas Edison makes the first recording of a human voice ("Mary had a little lamb") on the first tinfoil cylinder phonograph on Dec. 6.

**1884** Nipkow (from Germany) devises scanner for scanning and transmitting images.

**1878** Edison granted patent No. 200,521 on Feb. 19 for a phonograph using tinfoil cylinders, with 2–3 minute capacity.

**1884** George Eastman invents flexible photographic film.

# PART 6

# Games and Multimedia

## CHAPTERS



**1885**
Chichester Bell and Charles Tainter patent a rival "Graphophone" using wax-coated cylinders with vertical cut grooves.

**1887**
Bell and Tainter organize the American Graphophone Co. to make and sell the treadle-powered graphophone as a dictation device for businesses.

**1887**
Edison patents the motion picture camera.

**1888**
Oberlin Smith introduces basics of magnetic recording.

**1888**
"You push the button, we do the rest" is the marketing slogan for a new camera using film developed by George Eastman.

**1888**
Thomas Edison and William Kennedy Laurie Dickson attempt to record motion picture photos onto a wax cylinder.

**1888**
Emile Berliner patents the "Gramophone" using a flat 7-inch disk with lateral cut grooves on one side only, manually rotated at 70 rpm with 2-minute capacity.

**1889**
Thomas Edison and his assistant, William Kennedy Laurie Dickson, invent the kinetoscope, a device that lets them view moving pictures on film.

**1889**
Fusajiro Yamauchi founds a playing card company that will eventually become Nintendo.

**1889**
Growth is seen in sales of commercial cylinders and discs, mostly classical and Tin Pan Alley songs.

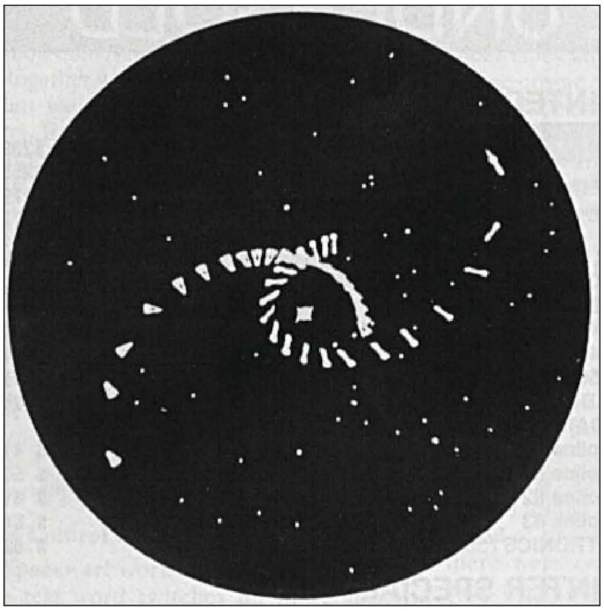
**1891-1895**
Dickson shoots numerous 15-second motion pictures using Edison's kinetograph.

**1895**
Edison markets the first spring-driven phonograph.

**1895**
First public demonstration of motion pictures displayed in France.

*If you could do whatever you wanted, it wouldn't be a game.*

—Clive Thompson



**First video game**

Developed at MIT, Spacewar made its debut in 1962. Here is the classic CBS Opening. The ships—one wedge-shaped and the other with a needle-nose—turn slightly away from the star and fire a short rocket blast (note the needle-ship's exhaust) to get into a comet-type orbit, and then rotate the other way to try shooting torpedoes at the opponent.

**THE** original IBM PC, compared to today's personal computers, was a poor, introverted little thing. It didn't speak, sing, or play the guitar. It didn't even display graphics well or show more than four colors at a time. Not only is today's multimedia revolution changing the ways we use PCs, it is also changing our use of information itself. Where information was formerly defined as columns of numbers or pages of text, we're communicating both to and from our PCs and using our voices, our ears, and our eyes, not simply to read, but to see pure visuals.

Today the distinctions among computers, movies, television, radio, CD players, DVD players, TiVo, and game consoles have all but disappeared. They are losing their individual identities to be co-opted, Borg-like, into one all-encompassing, networked system that serves up entertainment, information, and communication throughout the home and the work place. Add the cell phone, which is rapidly morphing into an extension of this computer/entertainment/communication personal conglomerate, and the boundaries of the home and office dissolve, too.

We are headed quickly toward the utopian idea of pervasive computing. Expect other devices to join the party. Sensors in the walls, in your bed, or in your cereal will be able to monitor your blood pressure, cholesterol, diet, and provide your doctor with a video view of your alimentary canal without you realizing any of it. As you play squash—prescribed because micro-analyzers in your vitamin pill radioed back a less than ideal body fat ratio—an ear bud whispers a warning that the stock you're watching shows signs of a collapse. Without missing a stroke, you whisper back to sell.

**1896** Gramophone improved with motor by Eldridge Johnson, who founded Victor Talking Machine Co. in 1901 with the "little nipper" dog as the trademark.

**1897** Development of the cathode ray tube by Ferdinand Braun.

**1898** Valdemar Poulsen patents the first magnetic recorder, called the "telegraphone," using steel wire.

**1900** Eastman Kodak Company sells its new Brownie box camera for $1.

**1901** Marconi sends a radio signal across the Atlantic.

**1901** Edison introduces Gold Mold cylinders for 50 cents each with improved hard wax surface and capability to be mass-produced by molding process. Victor Co. releases Red Seal 10-inch discs with 4-minute capacity for $1.00 each featuring famous European artists, such as tenor Enrico Caruso and baritone Mattia Battistini.

**1901** Phonograph discs made of hard resinous shellac sold.

**1902** Photoelectric scanning can send and receive a picture.

**1903** *London Daily Mirror* illustrates only with photographs.

**1904** Offset lithography becomes a commercial reality.

**1904** *The Great Train Robbery* creates demand for fiction movies.

**1904** The first comic book is printed.

**1904** The double-sided phonograph disc is invented.

**1904** A telephone answering machine is invented.

**1904** A photograph is transmitted by wire in Germany.

**1905** The first nickelodeon opens in Pittsburgh.

If all this turns out to be true, you can thank a bunch of computer hackers—an admirable term back then—who in 1961 had been set loose on a new PDP-1 computer in the laboratory basement at MIT. Without someone like them, computers would have very likely remained tools for crunching drab numbers and data, text-only machines that would have been only a glorified combination of typewriter, adding machine, and card catalog. But the MIT hackers nudged computing in the right direction to develop, after decades of growth, into talking, breathing, listening, living machines that are a part of our world and that invite us into their virtual worlds.


**Wolfenstein 3D**

The graphics were crude in the 1992 game, but the ability to move freely in a 3D world was exciting and a major step toward virtual reality.

After creating from scratch the basic software—compilers, debuggers, and text editors—needed by any computer to write real programs, the hackers plunged into creating a serious piece of software: *Spacewar*. The game began as cute graphics demonstration that allowed three points of light to interact with each other based on parameters entered at the keyboard. The three points of light quickly evolved into a star and a couple of spaceships, which soon developed the ability to fire torpedoes in the form of even smaller dots of lights. There was a realism to the game that most computer games for the next couple of decades didn't have. The objects followed laws of physics. The ships had to overcome inertia to get moving and overcome inertia to stop. The sun's gravity affected their paths and those of the torpedoes, and *Spacewar* allowed you to reverse the laws so the sun repelled objects instead of attracting them.

It wasn't until 1992 that computing took its next big step toward creating virtual worlds, although few back then could have seen where it was headed. id Software distributed a free game called *Wolfenstein 3D*. The back story was that you were a soldier on the loose in a Nazi prison. You had to fight your way through several levels full of enemy soldiers, vicious dogs, and the Big Boss himself.

The graphics were crude. But it had a grabber: It was modeled on a 3D environment that enabled you to move in any direction you wanted. Most games until then had

**1905** In France, Pathe colors black-and-white films by machine.

**1906** In Britain, a new process colors books cheaply.

**1906** Lee de Forest invents the three-element vacuum tube.

**1906** An animated cartoon film is produced.

**1906** Strowger invents automatic dial telephone switching.

**1907** The use of a cathode ray tube produces television images.

**1907** Bell and Howell develop a film projection system.

**1912** Motorized movie cameras replace hand cranks.

**1912** The U.S. passes a law to control radio stations.

**1913** Edison finally begins to sell flat Diamond-Disc players and recordings.

**1906** A program of voice and music is broadcast in the U.S.

**1906** Dunwoody and Pickard build a crystal-and-cat's-whisker radio.

**1906** Fessenden plays violin for startled ship wireless operators.

**1906** Victor sells the first Victrola with enclosed horn, but the name is later applied to all phonograph players designed as furniture.

**1907** In Russia, Rosing develops the theory of television.

**1911** Rotogravure aids magazine production of photos.

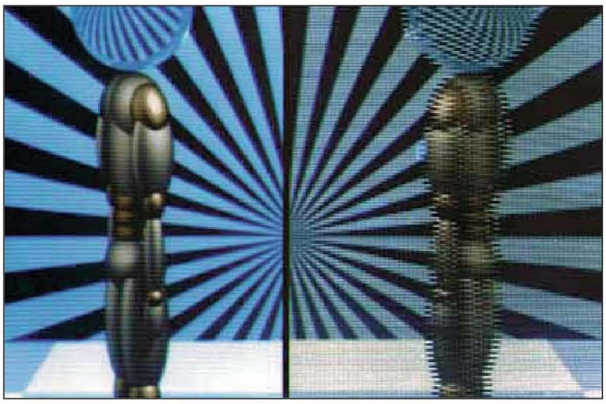
**1912** de Forest develops the Audion vacuum tube amplifier.

**1912** Feedback and heterodyne systems usher in modern radio.

**1914** ASCAP founded to enforce 1909 Copyright Act.

a predetermined course to follow. *Wolfenstein 3D* let you go and do what you wanted. The enemy soldiers had a primitive artificial intelligence that enabled them to react to what you did. (Fire your gun, for instance, and they might hear you and come running.)

*Wolfenstein 3D* was followed by id's *Doom*, which was *Wolfenstein 3D* on a big budget and with all imaginative shackles cast off. The popularity of *Doom* and all its imitators spurred the development of video cards and sound cards that could more realistically render a 3D environment, which made it possible for software developers and artists to create more and more realistic games until they reached the point today where each blade of grass and every strand of hair can move in individual reaction to a 3D world.



**1989 HDTV**

The first experimental high-definition display in 1989 illustrates the difference between it (on the left) and the interlace technology used by conventional television.
***Courtesy of Lucent Technologies***

Games have brought us to the point that our world and game worlds are separated by only a thin fabric of reality. It's easy to become caught up in these games and easy to see how they have already overlapped into television and movies, where some characters exist only as creatures of a computer performing next to flesh-and-blood actors.

We're in the next big revolution without knowing it. It's a revolution that enables a 3D model to contain an entire virtual world as a collection of software files, a database of objects, and their locations in the virtual world. Join in the 3D world through your computer, your TV, and maybe soon through your cell phone, and the database records your movements and your position and those of thousands of others in the same gameworld. It gathers information about what you should be seeing, hearing, and feeling and transforms that data into changes you see on the display; hear on your 3D, spatialized speakers; and feel in your force feedback controller. If it keeps on like this, soon, very soon, we might stay home for a visit to virtual reality more often than we leave the house to visit real reality.

**1914** In Germany, the 35mm still camera, a Leica, is invented.

**1915** Radio-telephone carries speech across the Atlantic.

**1915** Wireless radio service connects U.S. and Japan.

**1915** The electric loudspeaker is unveiled.

**1915** *Birth of a Nation* sets new movie standards.

**1916** Radios get tuners.

**1916** David Sarnoff envisions radio as "a household utility."

**1917** Frank Conrad builds a radio station, later KDKA.

**1917** Photo composition begins.

**1917** Condenser microphone aids broadcasting, recording.

**1919** People can now dial telephone numbers themselves.

**1919** Shortwave radio is invented.

**1920** Sound recording is done electrically.

**1920** KDKA in Pittsburgh inaugurates commercial radio.

**1922** Germany's UFA produces a film with an optical sound track.

**1922** *Nanook of the North* is the first documentary.

**1922** First 3D movie requires spectacles with one red and one green lens.

**1923** A picture, broken into dots, is sent by wire.

**1923** Russian-born engineer Vladimir Zworykin demonstrates his new invention, the iconoscope, which he claims will make it possible to transmit pictures (even moving pictures) through the air.

**1923** Kodak introduces home movie equipment.

**1923** First appearance of neon advertising signs.

**1924** Pictures are transmitted over telephone lines.

## KEY CONCEPTS

**3D graphics** Not the same as 3D movies, in which you have a sense of depth. Instead, computer animation is rendered in real time, in which you can infinitely change the viewpoint.

**AVI** Acronym for *audio/video interleave*, one of the most common file formats that combines video and sound.

**bitmaps** Graphics designed to look like skin, clothing, brick walls, and other 3D objects in games or virtual reality.

**DVI** Stands for Digital Visual Interface. It's a video connector standard that is designed for use with fully digital displays, such as those based on LCD and DLP technology.

**frame rate** Speed of animation, usually expressed in frames per second.

**laser** Originally an acronym for *light amplification by stimulated emission of radiation*, a laser is a device that produces a coherent beam of light. That is, the beam contains one or more extremely pure colors and remains parallel for long distances instead of spreading as light normally does.

**MIDI** Acronym for *musical instrument digital interface*, MIDI is a protocol for recording and playing back music on digital synthesizers supported by most sound cards. Rather than representing musical sound directly, it contains information about how music is produced. The resulting sound waves are generated from those already stored in a wavetable in the receiving instrument or sound card.

**MMORPG** Acronym for *massively multiplayer online role-playing game*, a computer role-playing game that enables thousands of players to play at the same time and interact with each other in an evolving virtual world over the Internet.

**MP3** An audio file compressed so that it's one-tenth the size of the original sound file. The compression technique is based on the third layer of MPEG, a scheme devised for compressing video as well as sound.

**rasterizer** Gaming software that translates the 3D geometry of 3D objects to a two-dimensional bitmap that can be displayed on the screen.

**real time strategy** Games involving military recreations of large battles. Players usually see the entire field and control many characters.

**RPG (Role Playing Game)** Long games with elaborate storylines that involve going on a quest and solving problems that increase the player's strength. For example, *Final Fantasy* and *Dungeons and Dragons*.

**shader** Plug-in code for graphics rendering software that defines the final surface properties of an object. Originally, shaders computed only surface shading, but the name stuck as new shaders were invented that had nothing to do with shading. For example, a shader can define the color, reflectivity, and translucency of a surface.

**shooter** Game devoted to shooting many monsters and characters. Typically, a **first-person shooter**, as in *Doom*.

**streaming** Sending video and audio transmission in real time over a network or the Internet.

**vector graphics** Images created by mathematically constructing a 3D framework out of **polygons**. The sides of the polygons are vectors in the sense that they have a length and a direction. The direction is set against some mathematical coordinate system that describes a 3D (or two-dimensional) space. Changing the mathematical description of the vectors animates the polygon images. **Shading** provides the polygons with various characteristics such as color, lighting, and texture to create surfaces for the polygons.

**virtual reality** The simulation of a real or imagined environment that can be experienced visually in the three dimensions of width, height, and depth. It can include other sensory experiences, including sound, touch, and feedback from "touched" objects, or other forces and sensations designed to enable a person to work in a computer environment by seeming to manipulate objects by handling them.

**1924** *The Eveready Hour* is the first sponsored radio program.

**1925** A moving image, the blades of a model windmill, is telecast.

**1927** Automatic Music Instrument Co. of Grand Rapids (AMI) introduces the all-electric coin-operated phonograph, the "juke box," to replace coin-operated pianos, but few are built before 1934.

**1928** Baird demonstrates color TV on electromechanical system.

**1929** In Germany, magnetic sound recording on plastic tape begins.

**1933** Singing telegrams are invented.

**1934** Drive-in movie theater opens in New Jersey.

**1939** Regular TV broadcasts begin.

**1948** Cable TV is installed.

**1958** Videotape delivers color.

**1924** There are two and a half million radio sets in the U.S.

**1925** All-electric phonograph is built.

**1926** Permanent radio network NBC is formed.

**1927** Talking films begin with Al Jolson in *The Jazz Singer*.

**1927** Philo Farnsworth invents a fully electronic TV.

**1928** Television sets are put in three homes and programming begins.

**1928** *Steamboat Willie* introduces Mickey Mouse.

**1930** AT&T tries the picture telephone.

**1933** Phonograph records go stereo.

**1936** Bell Labs invents a voice recognition machine.

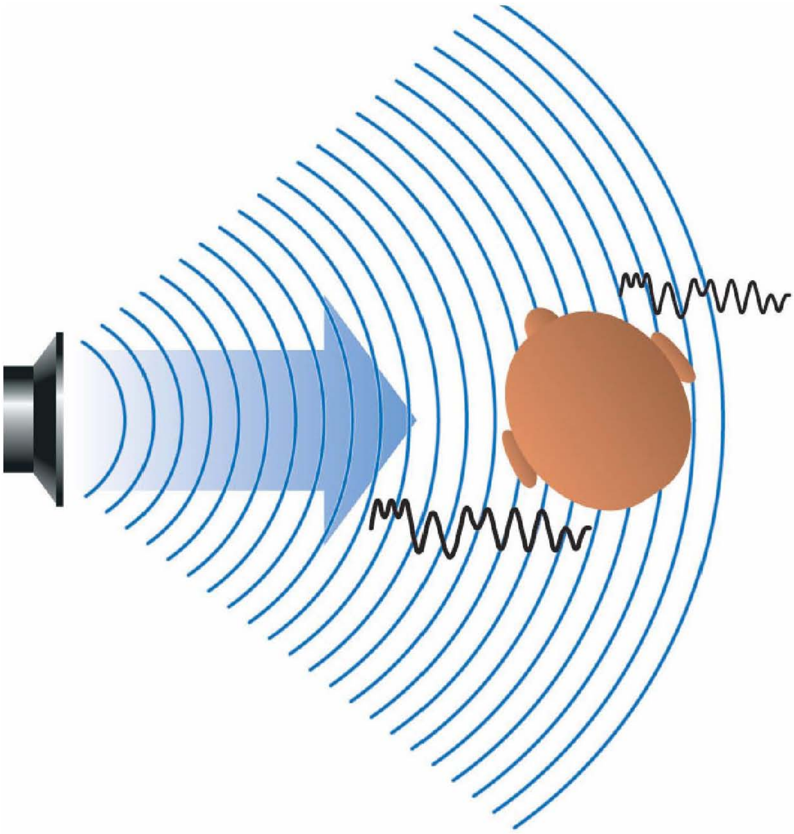
**1948** Land's Polaroid camera prints pictures in a minute.

**1954** Regular color TV broadcasts begin.

**1966** U.S. cars are equipped with 8-track stereo cartridge tape players.

CHAPTER 20

# How Multimedia Sound Works



**FOR** years, DOS and Windows personal computers sounded like cartoon roadrunners. They could play loud, high-pitched beeps and low-pitched beeps. But they were still only beeps. There was no hiding it.

We owe today's multimedia sound capabilities to game players. They saw the advantages of hearing realistic explosions, rocket blasts, gunshots, and mood-setting background music long before developers creating business software realized the practical advantages of sound. Now, you can listen to your PC speak instructions as you follow along on the keyboard, dictate a letter by talking into your PC, give your PC spoken commands, attach a voice message to a document, and not have to take your eyes off a hard-copy list while your PC sounds out the numbers as you're typing them into a spreadsheet.

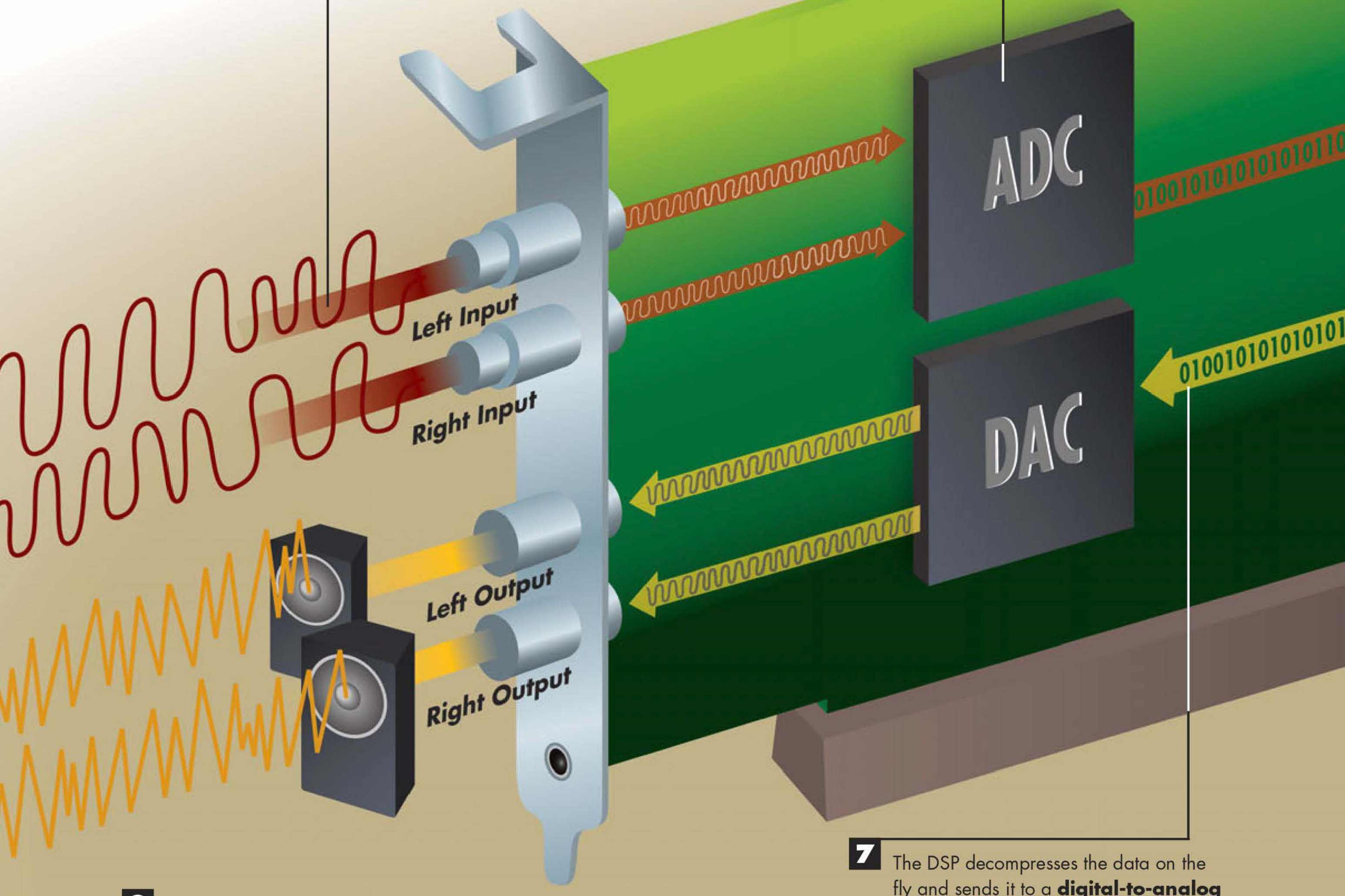
None of the multimedia that enhances business, personal, and family use of a PC could exist without sound capabilities. Multimedia CD-ROMs and DVDs bring their subjects to life in ways not possible in books, because you hear the actual sounds of whales, wars, and warblers, of sopranos, space blaster shots, and saxophones. Not that sound capabilities must always enlighten you on a topic. You should have fun with your PC, too. It won't make the work day shorter to replace Windows's error chime with Homer Simpson saying, "D'oh!" You won't be more productive every time a Windows program opens or closes if it makes a sound like those doors in Star Trek. And you'll spend more time than you should creating an MP3 song collection from your stockpiles of music CDs. But so what? Taking advantage of the sounds in a multimedia PC personalizes a machine that has a rap for being impersonal. Sound simply adds to the fun of using your computer. And we all spend too much time in front of these things for it not to be fun.

Lately, multimedia sound has taken a reverse spin. As we discussed in Chapter 14, now, instead of us listening to our computers, our computers can listen to us. Although a slow, painstaking version of voice recognition has been possible for years, it's only with the faster processing first made possible by the Pentium III and 4 processors and their technologies that natural speech recognition has become possible. We can now dictate, speaking in a normal voice, instead of typing. And although we do so much typing that it seems natural, if you think of it, there's hardly a more unnatural way to communicate than tapping little buttons. Don't throw away the keyboard just yet, but in the not-so-distant future, expect to be holding complete conversations with that machine on your desk.

# How Sound Cards Work

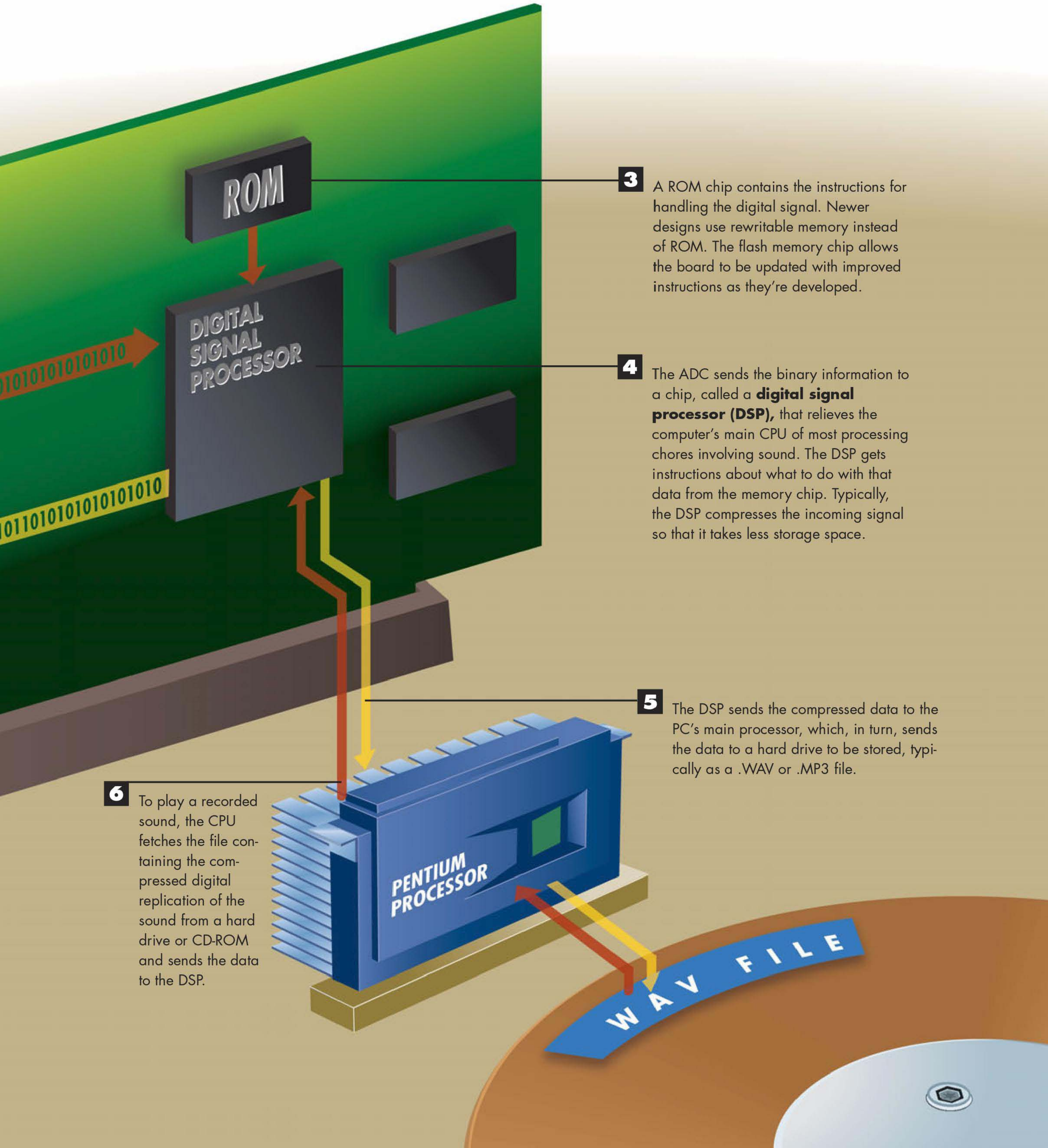
**2** The signals go to an **analog-to-digital converter (ADC)** chip. The chip changes the continuous analog signal into the 0s and 1s of digital data.

**1** From microphones or equipment such as an audio CD player, a sound card receives a sound in its native format—a continuous analog signal of a sound wave that contains frequencies and volumes that are constantly changing. The sound card can handle more than one signal at a time, allowing you to record sounds in stereo. For clarity, this illustration shows separate inputs and outputs for stereo, but left and right signals are almost always combined in a single jack.




**7** The DSP decompresses the data on the fly and sends it to a **digital-to-analog converter (DAC)** chip, which translates the digital information to a constantly wavering electrical current.

**8** The analog current is amplified, usually by an amplifier built into the PC's speakers. Then, the stronger current powers an electromagnet that's part of the speaker, causing the speaker's cone to vibrate, which re-creats the sound.



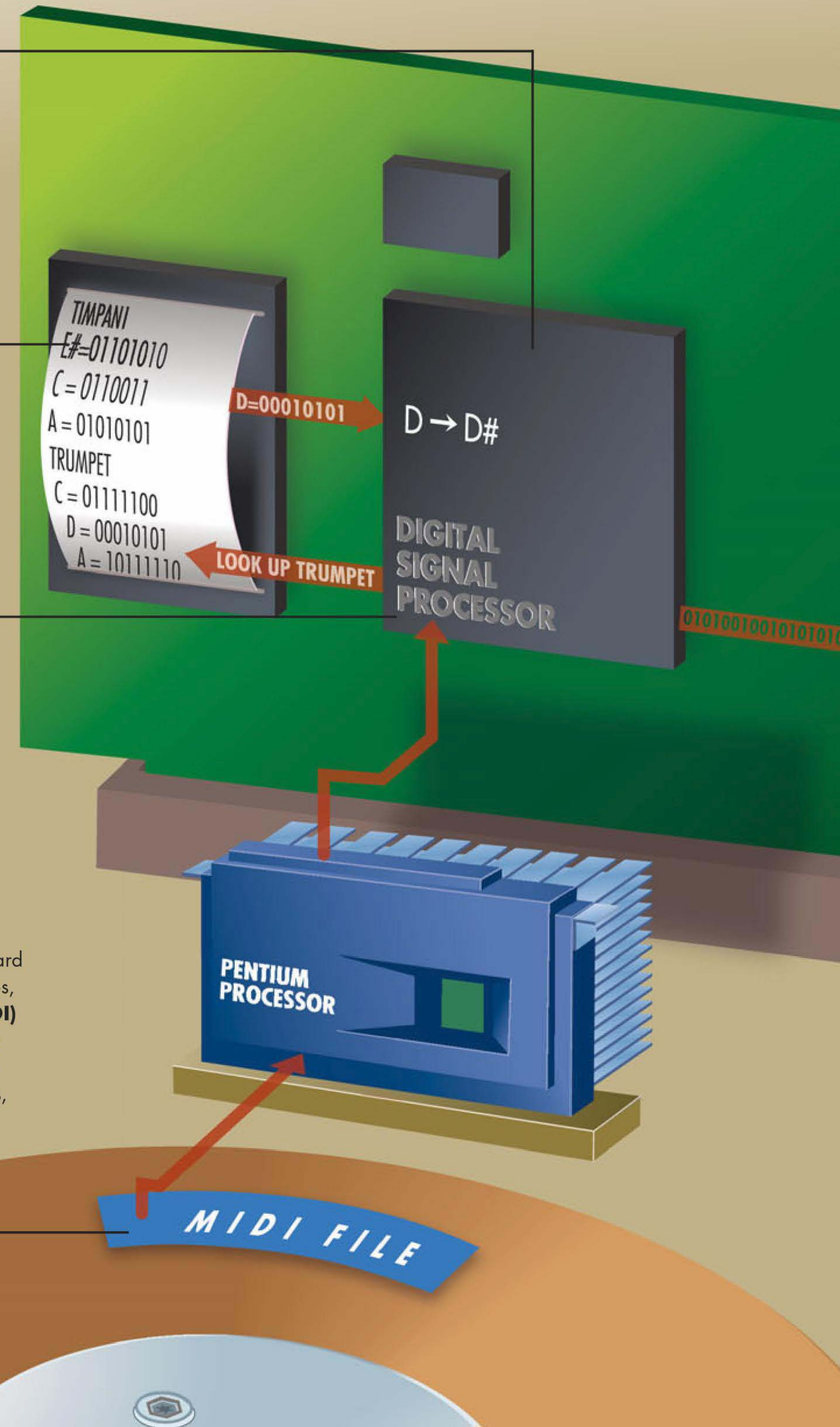

**3** A ROM chip contains the instructions for handling the digital signal. Newer designs use rewritable memory instead of ROM. The flash memory chip allows the board to be updated with improved instructions as they're developed.

**4** The ADC sends the binary information to a chip, called a **digital signal processor (DSP),** that relieves the computer's main CPU of most processing chores involving sound. The DSP gets instructions about what to do with that data from the memory chip. Typically, the DSP compresses the incoming signal so that it takes less storage space.

**5** The DSP sends the compressed data to the PC's main processor, which, in turn, sends the data to a hard drive to be stored, typically as a .WAV or .MP3 file.

**6** To play a recorded sound, the CPU fetches the file containing the compressed digital replication of the sound from a hard drive or CD-ROM and sends the data to the DSP.

# How MIDI and FM Synthesis Work

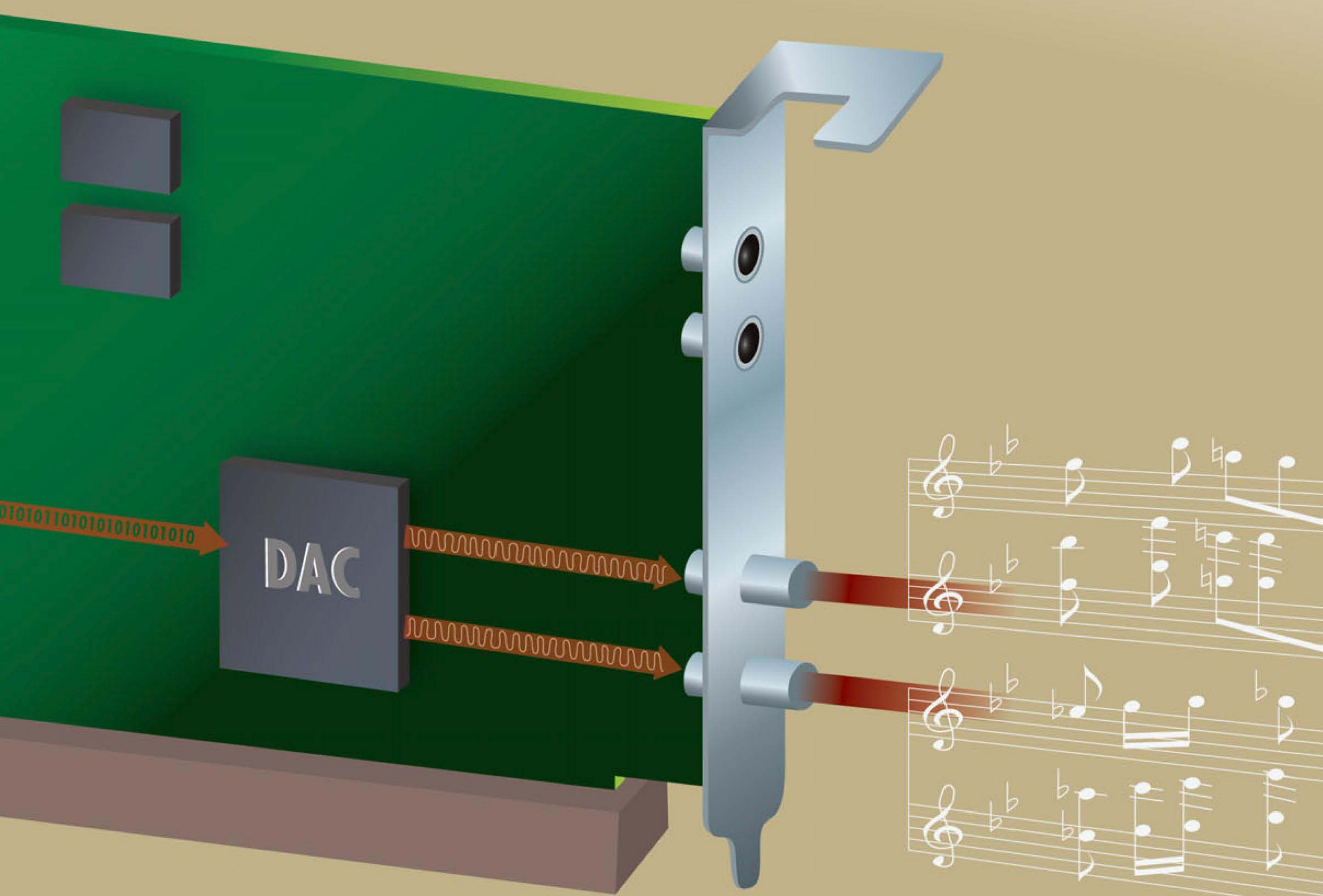



**1** Whereas some types of sounds are straightforward recordings, such as those contained in .WAV files, **musical instrument digital interface (MIDI)** sound was developed to conserve disk space by saving only instructions for how to play music on electronic versions of various musical instruments, not recordings of the actual sounds.

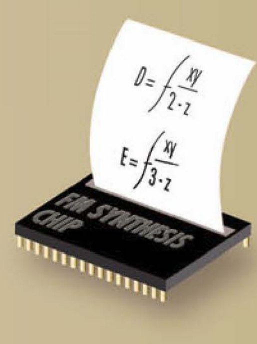
**2** The MIDI instructions tell the **digital signal processor (DSP)** which instruments to play and how to play them.

**3** If the sound card uses **wave-table synthesis** to reproduce the sound qualities of musical instruments, samples of the actual sounds different musical instruments make are stored in a ROM chip.

**4** The DSP looks up the sound in the ROM's table. If the instructions call for, say, a trumpet's D-sharp, but the table has a sample of only a D note for the trumpet, the DSP manipulates the sound sample to raise it to a D-sharp pitch.



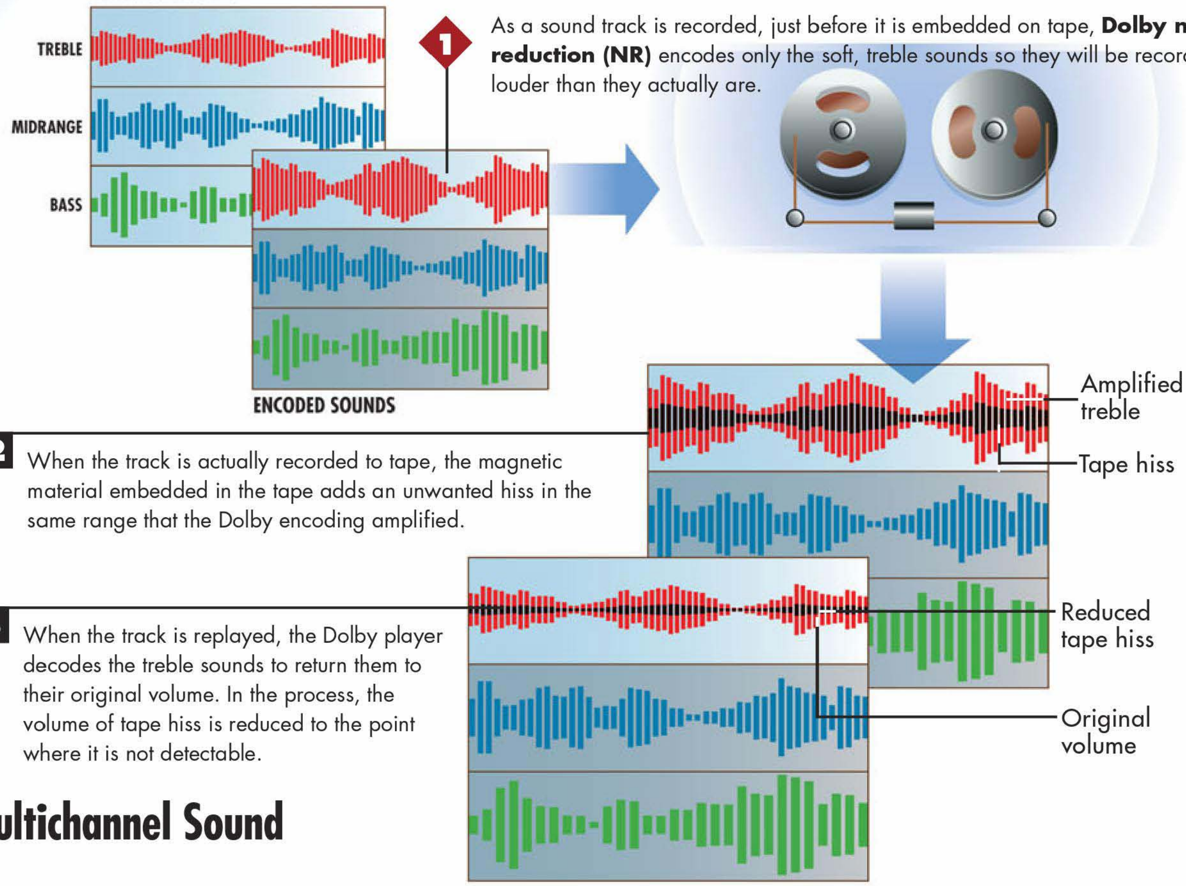

**5** If a sound card uses **FM synthesis** instead of wave-table synthesis, the DSP tells an FM synthesis chip to produce the note. The chip stores the characteristics of different musical instruments in a collection of mathematical descriptions called **algorithms**. By combining the DSP instructions with the algorithm, the chips synthesize a facsimile of the actual instrument playing the note. The chip handles some instruments better than others, but generally, FM synthesis is not as realistic as a MIDI wave table or .WAV sound reproduction.



# How Digital Sound Tricks Your Ear

## Dolby Noise Reduction

**1** As a sound track is recorded, just before it is embedded on tape, **Dolby noise reduction (NR)** encodes only the soft, treble sounds so they will be recorded louder than they actually are.

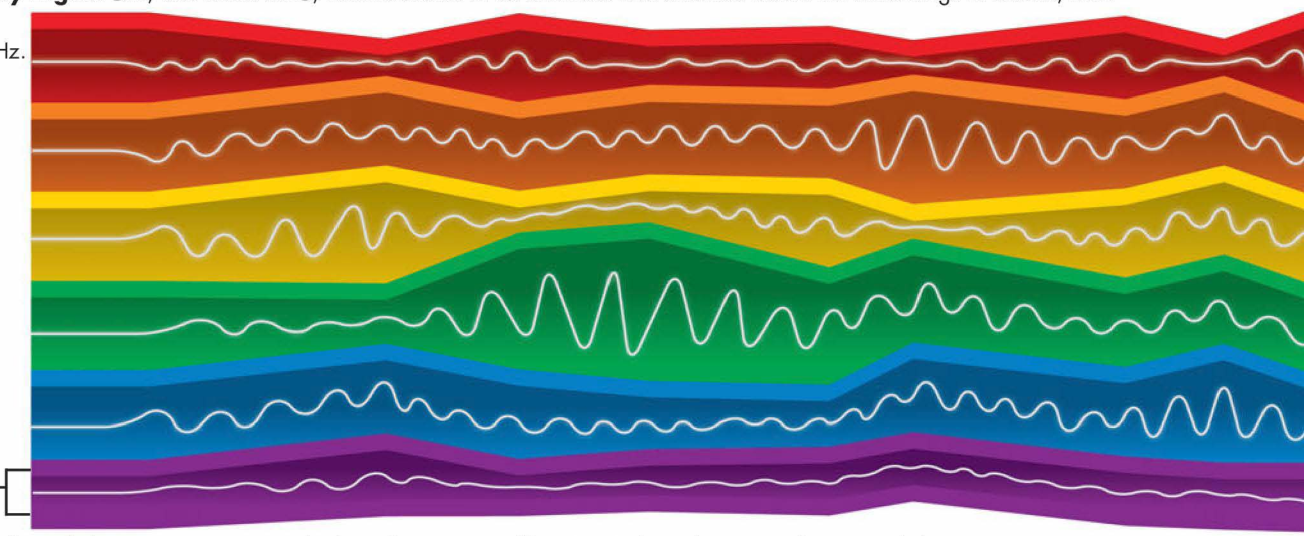



**2** When the track is actually recorded to tape, the magnetic material embedded in the tape adds an unwanted hiss in the same range that the Dolby encoding amplified.

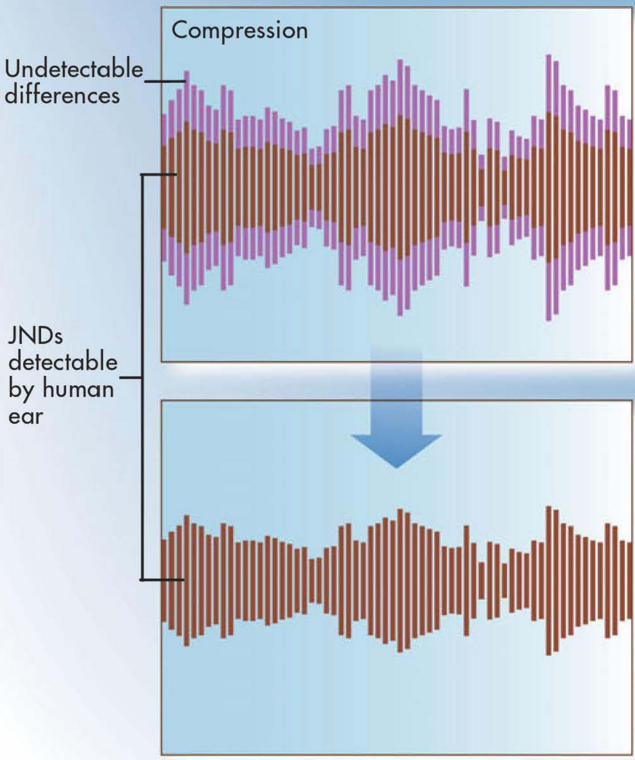
**3** When the track is replayed, the Dolby player decodes the treble sounds to return them to their original volume. In the process, the volume of tape hiss is reduced to the point where it is not detectable.

## Multichannel Sound

**1** Full **Dolby Digital 5.1**, also called **AC3**, records sounds in six channels. Five channels record the same range of sounds, from 3Hz to 20,000Hz.



**2** The sixth channel, the .1, is narrower. It's the **low frequency effect (LFE)** channel. It carries bass sounds from 3–120Hz, used for explosions, crashes, and similar loud sounds.



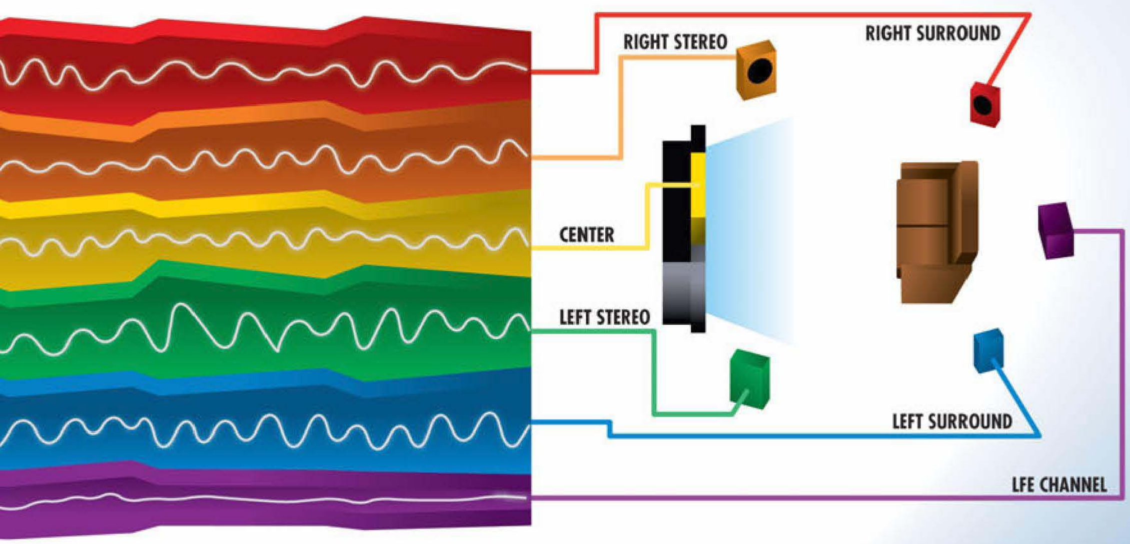

# MP3 and Digital Audio Compression

1 A stream of sounds—music, effects, or speech—contains changes in volume, pitch, and overtones that the human ear cannot distinguish because they do not exceed the **JND**—the **just noticeable difference**. If the intensity of a sound doubles, for example, the ear hears only a 25% increase; the extra increase is wasted on human ears. The JND is not a constant. The JND between any two sounds varies with the frequency, volume, and rate of change (which explains why you can listen to a car radio comfortably, but when you start the car the next time, the radio seems ridiculously loud).

2 MP3, Windows Media Audio, Dolby NR, and other forms of audio compression squeeze a sound file to as small as 10% of the original size by recording only the JNDs. It ignores differences the human ear can't distinguish. It also doesn't waste recording bandwidth by capturing sounds such as faint, high-pitched tinkles, which will be drowned out by the pounding of a bass drum.

3 Among the five main channels, Dolby bandwidth is based on each channel's needs. The center channel typically carries more data, and Dolby allots it a wider band of the bandwidth. But as the channels' relative requirements for bandwidth change, Dolby **dynamically**—on the fly—reallocates the sizes of all the channels to be sure the most data, and the most important data, gets through.

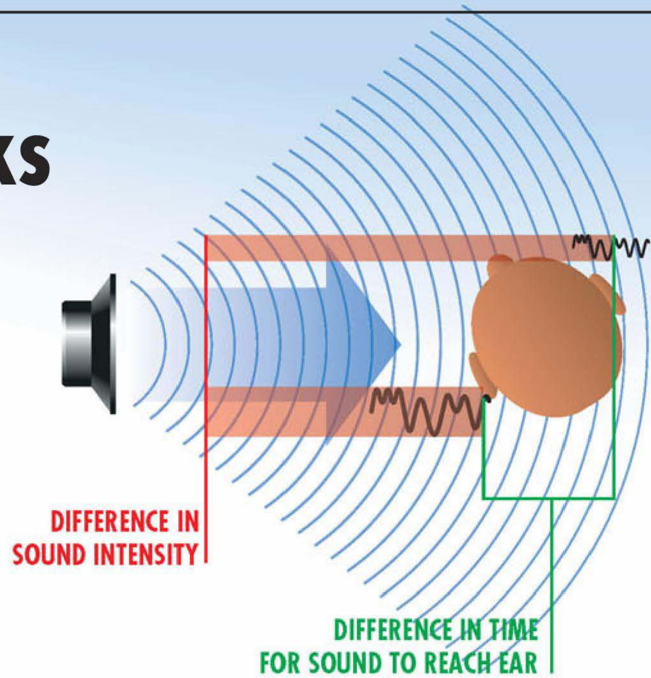
4 When replayed on a Dolby Digital 5.1 system, the sounds are separated along the six channels to individual speakers, typically three front speakers and two surround speakers to the sides. The sixth channel, with its explosive bass, goes to a nondirectional subwoofer that can be positioned anywhere. AC3 recordings can be played on systems that have only one, two, or four speakers. In that case, Dolby mixes the signals from the six channels as needed to create the most realistic sound it can for that system.

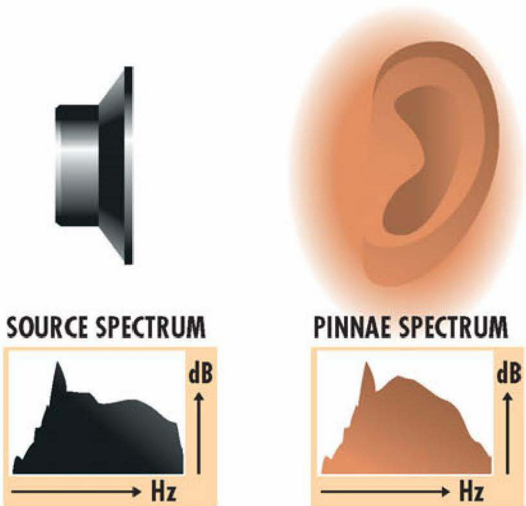



To understand how digital sound processing improves recorded sound quality and creates auditory illusions of space and environment, it's helpful to examine one of the oldest tricks in the digital audio repertoire, **Dolby noise reduction (NR)**.

# How 3D Audio Works

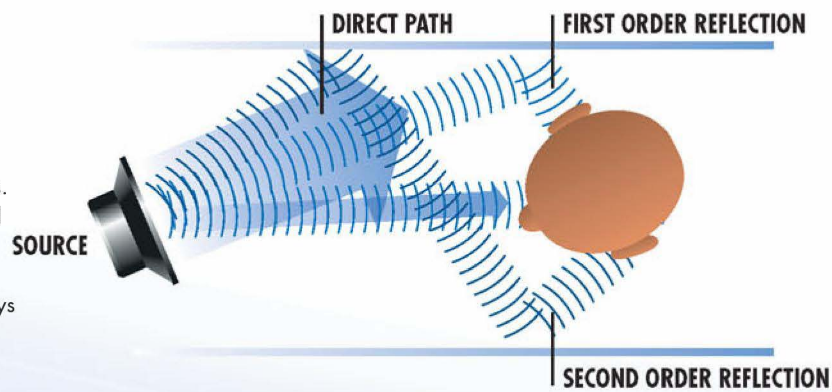
**1** One of the clues the brain uses to determine the source of a sound is the difference in how each ear hears the sound. An **interaural intensity difference** is detected because the sound is louder to the ear nearer to the source. An **interaural time difference** occurs because the sound reaches one ear sooner than the other ear.

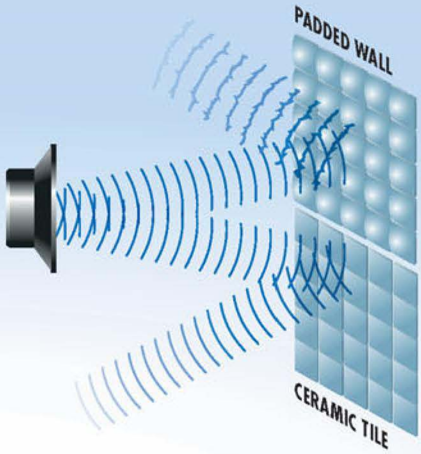


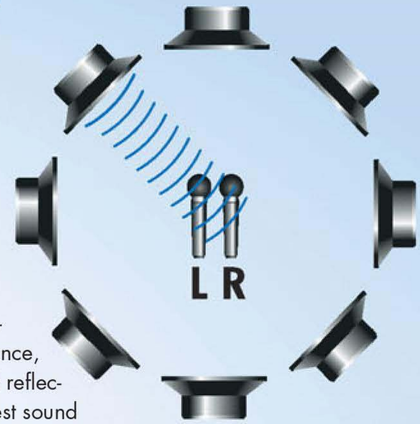

**2** Positional clues are also found in how the **pinnae**—the exterior ear flaps—deform the sound waves as they enter the ear. The contours of the pinnae echo, muffle, and accentuate differently the various components of a complex sound, depending on the angle with which the sounds strike the flaps.

**3** An important environmental factor is surrounding surfaces. They both reflect and absorb sound. The reflections called **reverberation**, or reverb, give clues to the direction a sound' is coming from and contribute to the realism of sound, because in the real world sounds are almost always reverberating off a variety of surfaces.



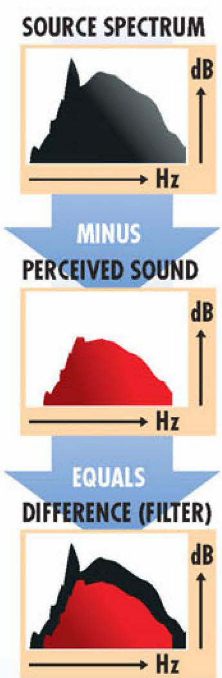



**4** The composition of the objects sound bounces off also influence its character. An echo in a padded cell sounds different than in a large tiled room.

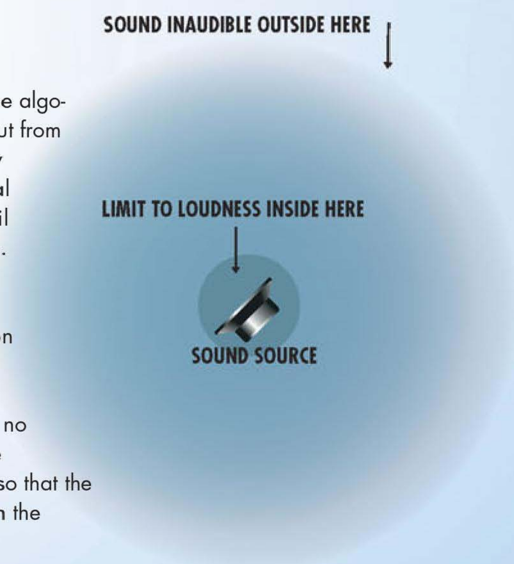



**5** Sound engineers capture the effect of distance, position, pinnae, and reflection by recording a test sound using microphones placed in a person's ear canals. As the sound source moves in a circle that has the person at its center, a record is made of the differences between the source sound and the perceived sound as recorded with the mikes. Similar comparisons are made between a source sound and the sound after it has bounced off an array of surfaces, from hard to soft, and smooth to rough.

**6** The values for the perceived sound are subtracted from the values of the original sound, creating a mathematical filter. The filters are used to create algorithms that software uses with **digital signal processors (DSP)** to create environmental sound that can be applied to any sound to re-create the illusion of a sound coming from a specific direction and reflecting off different surroundings.



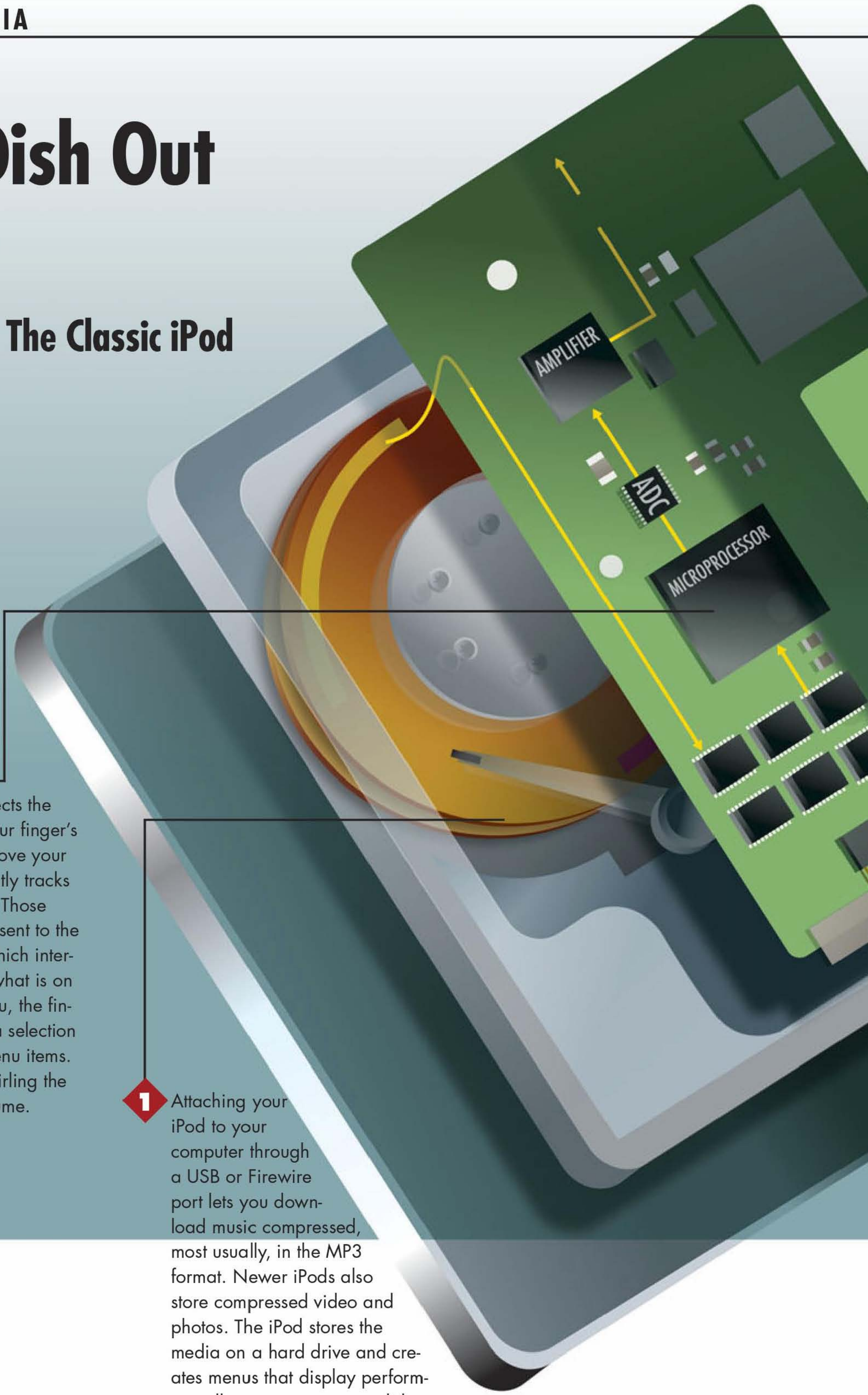

**7** In an immersive game, for example, the algorithm creates two spheres spreading out from each sound source that mathematically maps to a location in the game's virtual world. The sound cannot be heard until a virtual player enters the outer sphere. Inside the first sphere, the sound changes in volume and character as the player's position changes in relation to the source and to surrounding surfaces. The inner sphere represents a space where the sound gets no louder no matter how close the player gets to the source. The inner sphere is necessary so that the volume doesn't increase infinitely when the player is right on top of the source.



# How iPods Dish Out Media To Go

The Apple iPod was not the first MP3 player, and you could argue that it does a better job of buying songs than playing them. But it is nevertheless the undisputed ruler of portable music—and video. It owes its success to an inspired design that banishes obvious buttons, dials, and knobs to replace them with stealth controls that create only the tiniest ripples in the smooth surface of the player. An early version had four subtle buttons beneath the iPod's LCD screen, but even they were too obtrusive for Apple's designers, who moved them to the **clicker dial** so that one simple circle is the only control you need.

## The Classic iPod




**4** The controller chip detects the increased charge at your finger's position, and as you move your finger, the chip constantly tracks its position and speed. Those two measurements are sent to the iPod microprocessor which interprets them in terms of what is on the screen. If it's a menu, the finger movement causes a selection bar to move among menu items. If a song is playing, twirling the wheel changes the volume.

**1** Attaching your iPod to your computer through a USB or Firewire port lets you download music compressed, most usually, in the MP3 format. Newer iPods also store compressed video and photos. The iPod stores the media on a hard drive and creates menus that display performers, album names, songs, slide shows, and video choices on the screen.


**2** You make menu selections and control playback functions such as volume, pause, and skip by using the Click Wheel. Four labels along the wheel's rim and a raised, but unlabeled, area at the center rest on top of five buttons. The four rim buttons, which click when pressed, are for functions such as play/pause and stop. The center button selects whatever is highlighted on a menu.

**3** On the undersurface of the Click Wheel is a metallic grid that holds a faint electrical charge. Each intersection in the grid is mapped to a different address by the wheel's controller chip. When you run your finger around the rim, even though you exert no pressure against the wheel, your finger acts as a potential electrical **ground** and attracts the grid's charge.


## The Shuffle, Nano and Touch

Despite the iPod's small size and weight, Apple has introduced two more iPods that carry compactness to an extreme by eliminating the hard drive and storing all music tracks in non-volatile memory.

**The Shuffle** weighs less than an ounce and measures only about 1 × 1.6 inches. It has no LCD screen and stores about 240 songs in 1GB of memory.

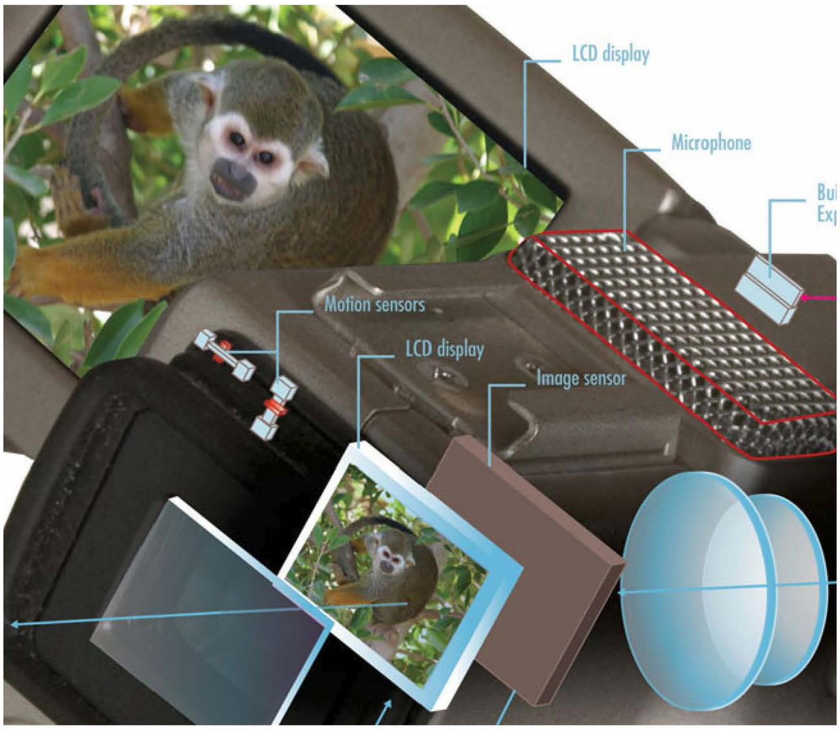
**The Nano** has a color LCD screen and 4 or 8GB of memory, and yet it is considerably smaller and lighter than a classic iPod. It stores about 1,000 songs for each 4GB of memory.

**The Touch** The resemblance of the iPod Touch to an iPhone is no coincidence. Despite the popularity of Apple's Click Wheel for the classic iPod, Apple chose to go with the **multi-touch** technology developed for its cellular phone. (See "How the iPhone Works," p. **358**, for a detailed explanation of multi-touch.) Like the iPhone, the touch displays video and plays MP3s. All it lacks to make it an iPhone is a phone. One thing it does have that the iPhone doesn't is built-in WiFi for wireless connections to Apple's iTunes website, allowing you to buy and download songs directly to your iPod.

**5** When you select a song or photo, the iPod copies that file from its permanent storage on the hard drive to a temporary storage area in RAM and turns off the hard drive to conserve the battery. Longer music or video is fed to RAM in chunks and the drive turned off between transfers.

CHAPTER 21

# How Multimedia Video Works



**VIDEO** is nothing new. We grew up with Howdy Doody, Gilligan, and Teletubbies. In many households, the camcorder has replaced the still camera as the memory-catcher of choice. So, why does video's arrival on PCs seem like such a big deal? It's precisely because video in more traditional forms has become so much a part of our lives. We rely daily on talking, moving pictures to get so much of the information we need to learn, to conduct business, and to lead our personal lives.

All the excitement and technical innovations in multimedia concentrate on video and audio, which has it all backward. The excitement isn't really in what video brings to computers. It's what computers bring to video. After all, we've had multimedia ever since the first talkie (movie with sound). VCRs have been around a lot longer than DVDs. So, what's the difference between a videotape and a DVD? What's the big deal?

Videotape doesn't have **random access**. The freedom to move to any point in a stream of information is random access. It's where RAM gets its name, **random access memory**. Early computers used magnetic tapes to store programs and data, and using them was slow because to get to where data Z is stored, you have to go past data A through data Y. That's **sequential access**.

It's exactly this issue of access that differentiates multimedia video from a videotape. You have control over what you see and hear. Instead of following a preprogrammed course of videos and animation, you can skip about as you like, accessing those parts of a multimedia program that interest you most. Or, with videoconferencing, you can interact live with another person in a different part of the world and work on the same document or graphic simultaneously.

Despite the superficial resemblance between a TV set and a PC display, the two produce an image in different ways. At least before high-definition digital TV started to become more common, television has been an analog device that gets its information from continuously varying broadcasting waves. A traditional computer monitor uses analog current to control the image, but the data for what to display comes from digital information—zeros and ones. Of course, most LCD panels don't even bother with that, using a purely digital connection from PC to display.

The flood of data can easily exceed what a display can handle. That's why multimedia video is sometimes small and jerky. A smaller image means less information—literally, fewer pixels—the PC has to track and update. The jerkiness comes from the slow update of the image—only 5 to 15 frames per second (fps), compared to 30 fps for a TV or movie. By further increasing data compression, many of these limitations have been all but eliminated. MPEG compression, for example, lets multimedia video cover the entire screen at a full 60 fps. Further development of the techniques described here for compressing and transmitting will eventually make computer video as ubiquitous as sitcom reruns.

# How the Digital Camcorder Captures Future Memories

The digital camcorder is not your father's 8mm movie camera. Fifty years ago, family events and vacations were recorded on narrow silent film that faded with the years and grew jerkier each time it was fed through the home projector. Today, the family video camera comes within a hair-breadth of producing movies—videos that are technically as good as you see in the theaters or on a TV set. For all the similarities between digital video and still cameras, they are, for the time being, different animals. Because the subjects in videos are constantly moving, the eye doesn't notice if the image is not as sharp as a good photo. But the two animals are evolving into a new creature—one joins the different circuitry and mechanisms needed to shoot both stills and motion.

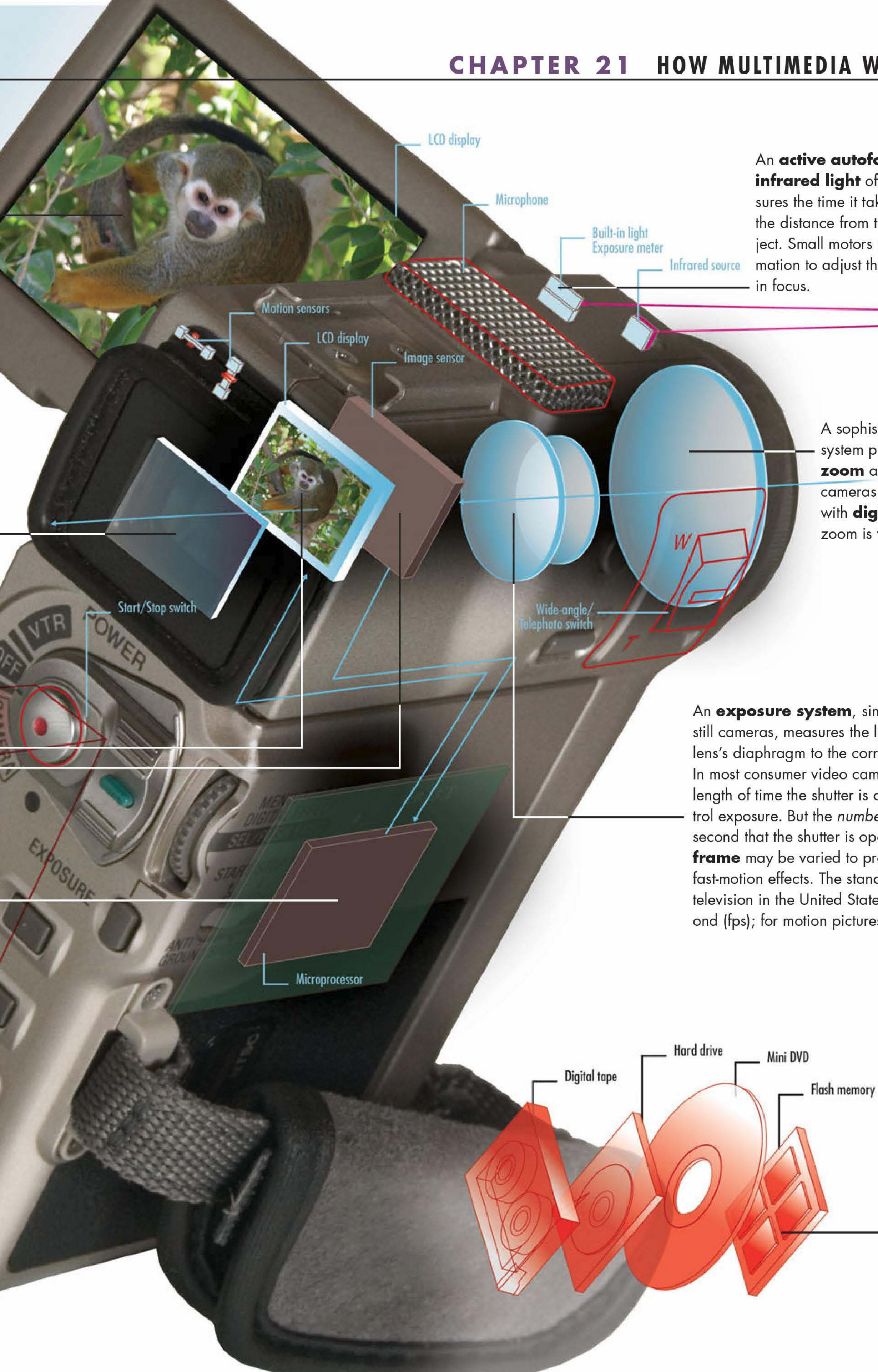
The operator of a low to mid-range digital camcorder has a choice of equipment for framing a shot. A **viewfinder** with a magnifying glass in the end of it enlarges a small LCD image that shows the lens's vantage point, focus, and amount of zoom, regardless of whether the camera is recording. The method cuts out distraction and lets the photographer concentrate better on the subject.

The second choice is a larger LCD screen that can be twisted to any position, even toward the front of the camera so the photographer can be in the **frame** while keeping an eye on what the camera is seeing.

The focused light from the lens falls on an **image sensor**, called a **charge-coupled device (CCD)**, or **complementary metal-oxide semiconductor (CMOS)** chip. Over the surface of the sensor lie thousands of **photodiodes**, or **pixels**, each of which generates electrical current in proportion to the brightness of the light that strikes it. Note that camcorders use fewer pixels than still cameras, which count their pixels in the millions. Camcorders are more concerned with hustling images through the system, at rates of a dozen or two each second, than with image sharpness. In fact, the millions of pixels in a camera would quickly be bottlenecked by the relative slowness with which camcorders can store the visual information they gather.

The image sensor in less expensive camcorders is larger than the actual image size. This is so the camera can use **electronic stabilization** to compensate for any jiggle or vibration that would otherwise be captured by the camera. Small gyroscopes in the camera detect the movements and signal the camera's circuitry. If the movement, for example, shifts the image hitting the sensor to the right, the circuitry shifts which pixels are being recorded to the right. Note that the pixels do not actually move, but rather the responsibility each has to capture a certain part of the image moves to a different pixel. More expensive camcorders make use of optical image stabilization to move the actual image sensor.

After an **image processor** in the camera massages the visual data passed on by the sensor, the processor converts the image to a system of numerical values that represent different colors and hues. They are joined to similar digital information coming from the camera's microphone. The combined signal is recorded, usually to magnetic tape.




An **active autofocus** system bounces **infrared light** off the subject and measures the time it takes to return to gauge the distance from the camera to the subject. Small motors use the distance information to adjust the lens so the subject is in focus.

A sophisticated zoom lens system provides real, **optical zoom** as strong as 10x. Some cameras try to embellish that with **digital zoom**, but optical zoom is what really counts.

An **exposure system**, similar to those found in still cameras, measures the lighting and adjusts the lens's diaphragm to the correct aperture opening. In most consumer video cameras, varying the length of time the shutter is open is not used to control exposure. But the *number* of times during each second that the shutter is opened to create a **frame** may be varied to produce slow-motion and fast-motion effects. The standard for commercial television in the United States is 30 frames per second (fps); for motion pictures, 24fps.

Some cameras provide an option of recording to solid-state flash memory, similar to that used in still cameras. Others record the video and sound directly to a DVD or a hard disk drive.

# How a Digital Camera Squeezes Video Down to Size

Compressing one picture on-the-fly is daunting enough for a digital still camera. But consider the task that confronts a video camera: compressing 15 to 30 images each second—and putting the squeeze on sound at the same time. There is no time for lossless compression, even if it could do a good enough job to store even a short feature on a DVD. The technology universally used for professional and amateur video is a lossy compression called MPEG, named after the organization that created it: the Motion Picture Experts Group. MPEG compression is likely to be used on anything you watch on TV, your computer, or iPod.



**1** An essential part of video compression strategy is limiting the amount of work the video camera has to do to record so many frames each second the camera is rolling. Standard TV (non–high definition with a 4:3 aspect ratio) is made up of 720 pixels across and 576 pixels vertically. Depending on the sophistication of a digital video camera, frames may be limited to 640 pixels wide and 480 pixels high, or even as few as 352 × 288. At 640 × 480, the camera is relieved of the task of recording information for a hundred thousand pixels for every frame it processes.


**2** Inside a video camera, you'll find a chip that specializes in MPEG compression (or circuitry that does the same job built into the master microprocessor). Part of the job MPEG performs is similar to the lossy JPEG compression used to shrink still photos. MPEG looks for redundant data. The wall in the background of a video may be painted a solid red, making it unnecessary for a video camera to record the same information about each of the red pixels that make up the wall. As JPEG compression did with a blue sky in the example a few pages previous, MPEG establishes reference values for the wall's shade of red and refers all the pixels that make up the wall to that color.

**3** Of course, few things in the real world are consistently the same color. Vagaries of light and shade cause a spot in the background to be a slightly different shade. MPEG makes a decision: Will anyone notice if it makes that errant shadow the same color as the rest of the wall? If the decision is "no," that's a few thousand more bits saved.

**4** One compression technique assumes, rightly, that much video changes little from one frame to another. To save space recording the video—and broadcasting it, for that matter—the technique combines part of one frame with part of the next one. In the case where there is little change, the economy works.

But in this case, for example, where the scene changes abruptly from one frame to the other, the technique results in a fleeting visual glitch.

**5** The same premise is behind **motion estimation**, which assumes that consecutive frames will be the same except for changes caused by objects moving between frames. By examining the changes in two of the frames, MPEG predicts the position of objects in a third frame.

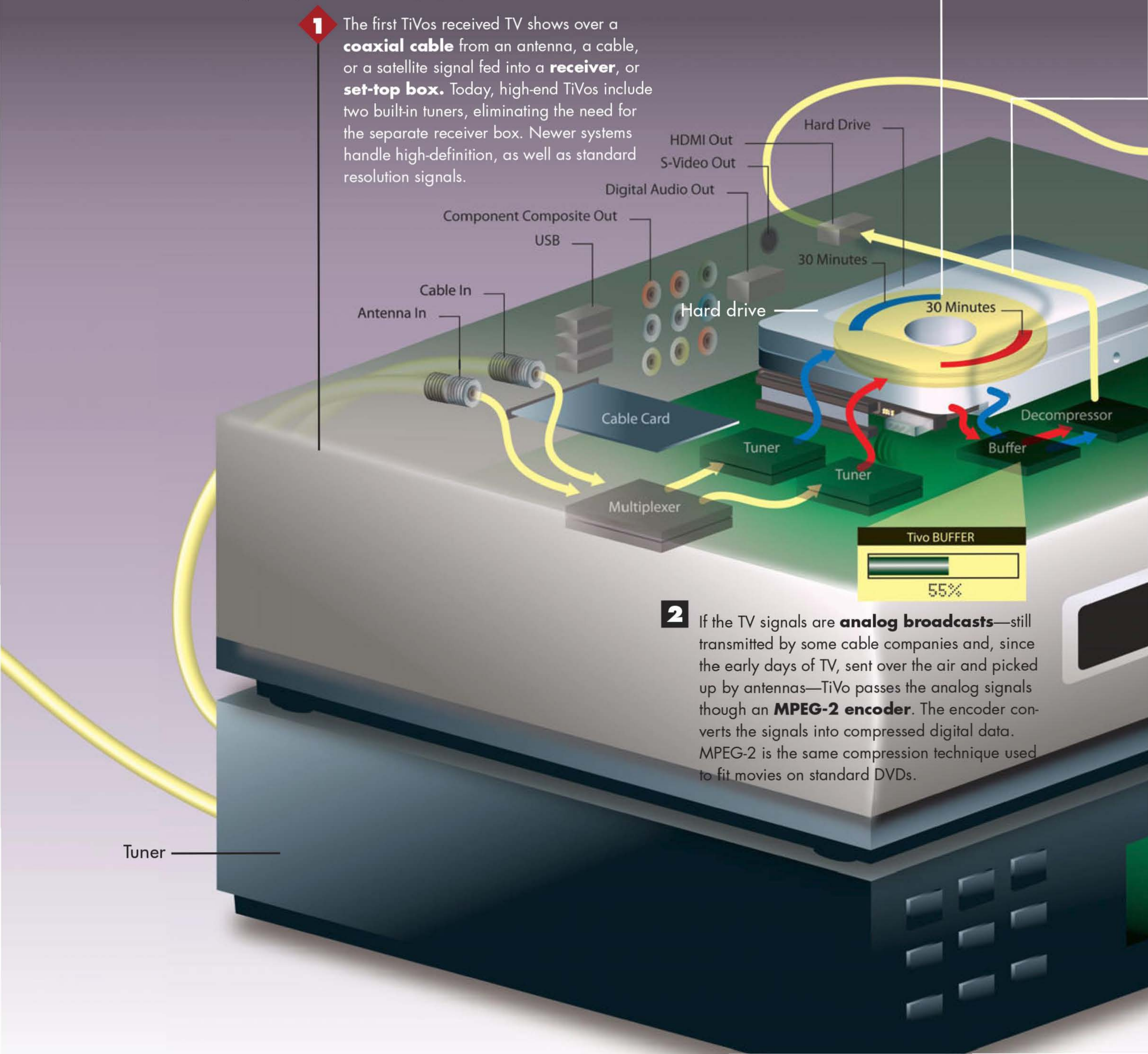
**6** **Key frame** compression records only certain crucial frames and deduces from them what the missing intervening frames should look like.

**7** A similar technique makes a record of one frame that is largely unchanged in subsequent frames. The only data recorded is the **delta**—the differences that occur in the frames trailing the original reference frame.

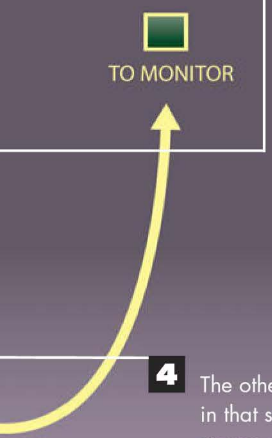
# How TiVo Unites TVs and Computers

A TiVo is a computer specialized for recording and replaying television programming. (*TiVo* is the brand name for one popular brand of **digital video recorder**—**DVR**—but the name is used so broadly that it might become genericized, going the way of *escalator*, *aspirin*, *zipper*, and *popsicle*.) By any name, the technology has all but replaced the tape-based VCR the way MP3 players have replaced portable audio tape players, like Sony's Walkman.

**1** The first TiVos received TV shows over a **coaxial cable** from an antenna, a cable, or a satellite signal fed into a **receiver**, or **set-top box.** Today, high-end TiVos include two built-in tuners, eliminating the need for the separate receiver box. Newer systems handle high-definition, as well as standard resolution signals.




**2** If the TV signals are **analog broadcasts**—still transmitted by some cable companies and, since the early days of TV, sent over the air and picked up by antennas—TiVo passes the analog signals though an **MPEG-2 encoder**. The encoder converts the signals into compressed digital data. MPEG-2 is the same compression technique used to fit movies on standard DVDs.




**3** TiVo splits the digital signal into two duplicate data streams. One stream is sent to the TV for immediate viewing. If the TV is hi-def, TiVo sends the signal in digital format through a **high-definition multimedia interface (HDMI)**, a connection with enough **bandwidth** to carry the HD signals and sound on a single cable. For a conventional TV, however, the signal first passes through an **MPEG-2 decoder**, which converts it back to the analog signals that type of TV requires.


**4** The other data stream remains digital and goes to a hard drive operating under the Linux operating system. If the TV show contained in that stream is one you've asked TiVo to record, the entire file is written to the disk and saved there for as long as you've programmed TiVo to keep it. The image quality and the number of hours the drive can hold depend on how much compression the MPEG-2 encoder applies to the signal. Basic quality uses .25-.35 gigabytes (GB) of storage for each hour. High quality, non-high definition uses about 3.5GB per hour. HDTV uses nearly 9GB for each hour.


**5** TiVo records programs without being asked. It maintains records of programs you watch and those you've given a thumbs-up using the remote. Based on those programs' categories—such as news or comedy,their performers and subject matter—TiVo examines the schedule for two weeks of programming downloaded every night and, when it's not completely busy recording shows you've ordered, it records other programs that appear to be a match to the shows you prefer. The more you use TiVo, the more accurate its picks become.


**6** At all times, TiVo also uses the hard drive to record a half-hour **buffer** of whatever is showing on the TV since the channel was last changed. The buffer lets you pause a show or replay anything the TV has shown in the last 30 minutes (provided the channels have not changed). Pressing the remote's pause button makes TiVo freeze the current frame while it continues to record the program. For a replay, TiVo finds the appropriate **time marker** in the buffer and repeats the program from that point.

CHAPTER 22

# How Game Hardware Puts You In the Action

**TAP-CLICK.** Tap-click. Tap-click. Most of the time… Tap-click… using your PC isn't the most exciting pastime. Tap-click. Tap-click. Not in a visceral sense. Tap-click…. The only things you get your hands on are the keyboard… Tap… and mouse… Click…. And no matter how their manufacturers dress them up with lights and shapes that look like a Calder sculpture, they're still unplayful assemblies of plastic that you can tap… and click. Usually about the most exciting thing you'll encounter…. Tap-click… Tap, tap. Click, click… is a rollicking case of carpel-tunnel syndrome. Tap-tappity tap. Clickity-click.

What you really want in your hands—at least if you're male—is a weapon of some kind. Something that instills and announces the rough, no-nonsense type of hombre you are. An office typist? You better watch your mouth. You're a soldier, pilot, samurai, bombardier, sheriff, clutch hitter, stock car driver, swordsman, 3-point shot specialist, martial artist, sportsman, fullback, pool shark, and sniper—at least if you have the right game hardware.

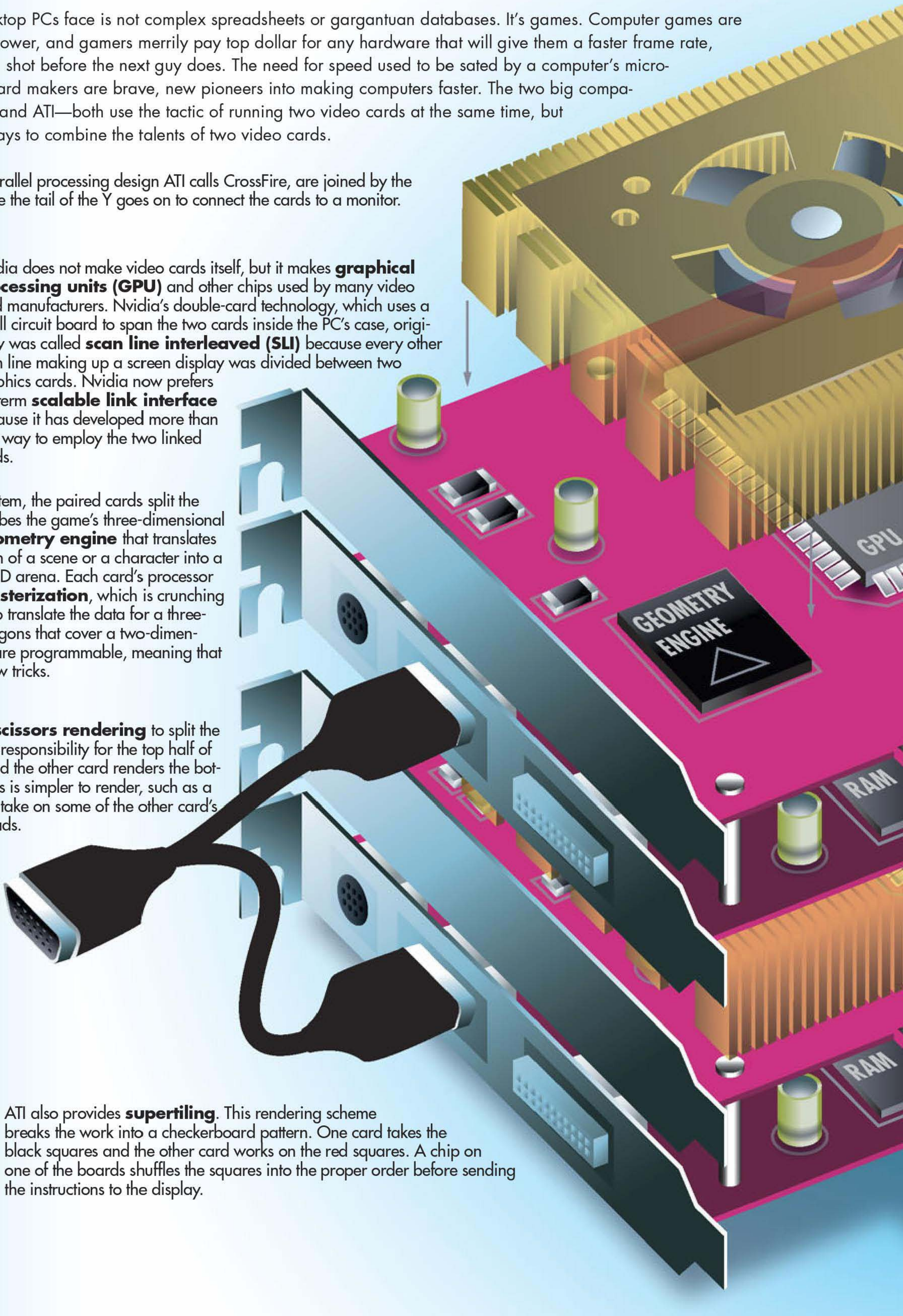
There was a time when people thought it was a good idea to control the movements of a computer game character with the W, A, S, and D keys, and admittedly, many still do. It shows not just how desperate they were for computer games and hardware yet to be invented, but how the idea of using a computer to play games is one of those inborn instincts, like the natural, primal desires to cook meals over an open fire and to avoid mimes.

Still, it took a while for the computer industry to produce decent game hardware because doing game controllers right can be expensive. The joystick that controls a fighter plane in *Flight Simulator* must be as precise and capable at the stick in a real jet. There are now controllers that emulate the functions of throttles, steering wheels, gas and brake pedals, fishing rods, golf clubs, bats, and pistols, all in manners so realistic—push some of them, and they push back—that you can easily forget it's all a game.

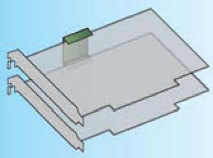
We'll see here how they work, from the classic joystick through the two-fisted game controls that have become ubiquitous and caused young boys to memorize obscure combinations of buttons to push so their onscreen game charter will tear off their opponent's head. And we'll get inside Nintendo's revolutionary Wii Remote—the Wiimote—that destroys the barrier between a game and playing it with natural movements.

# How Video Cards Break the Game Barrier

The most serious challenge desktop PCs face is not complex spreadsheets or gargantuan databases. It's games. Computer games are voracious users of processing power, and gamers merrily pay top dollar for any hardware that will give them a faster frame rate, which means they can get off a shot before the next guy does. The need for speed used to be sated by a computer's microprocessor, but now the video card makers are brave, new pioneers into making computers faster. The two big companies in gaming video—Nvidia and ATI—both use the tactic of running two video cards at the same time, but they've each found different ways to combine the talents of two video cards.



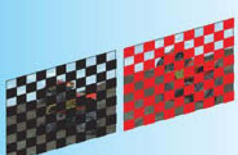

**1** The two ATI cards, in a parallel processing design ATI calls CrossFire, are joined by the cables that form a Y before the tail of the Y goes on to connect the cards to a monitor.



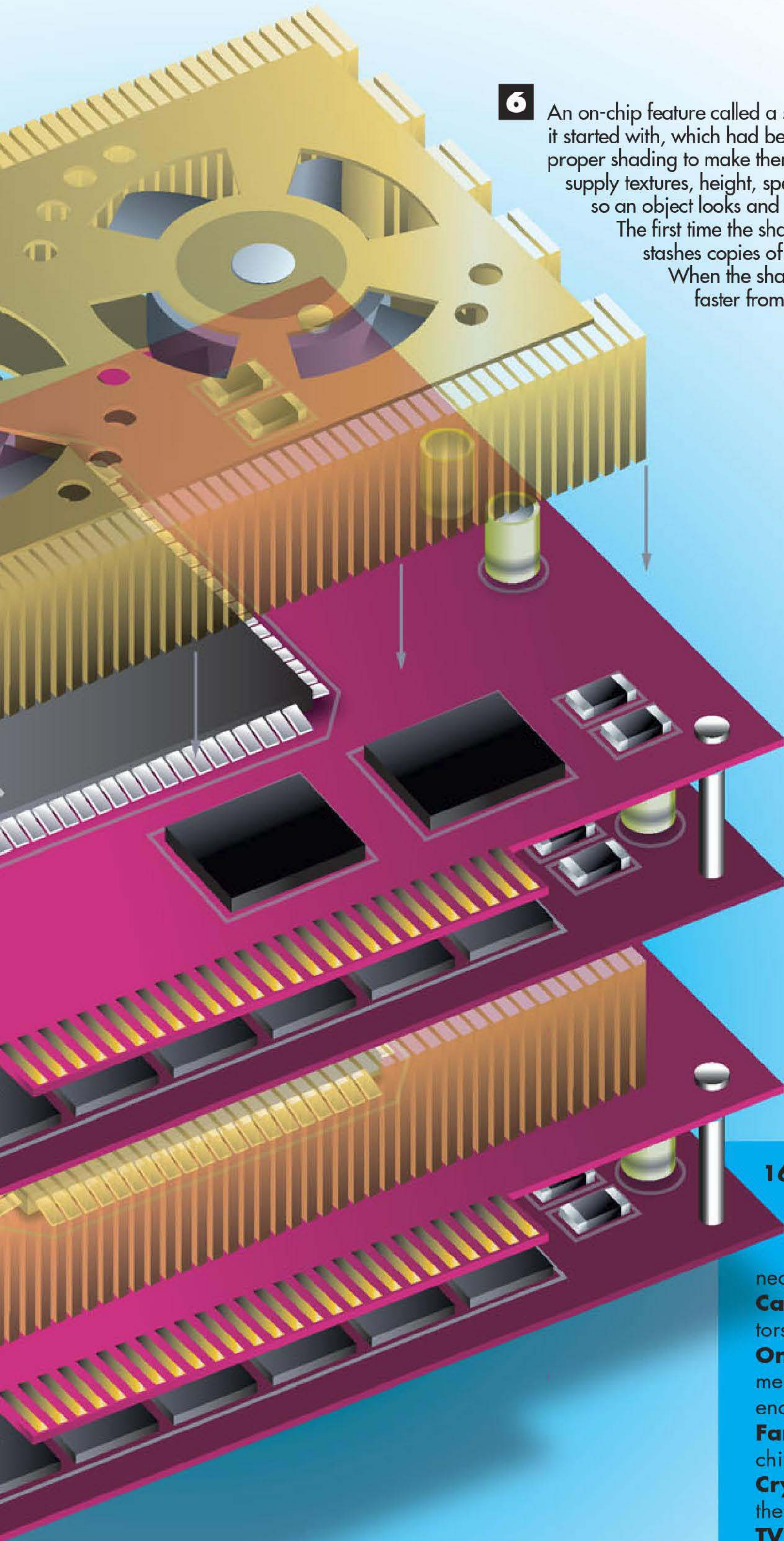
**2** Nvidia does not make video cards itself, but it makes **graphical processing units (GPU)** and other chips used by many video card manufacturers. Nvidia's double-card technology, which uses a small circuit board to span the two cards inside the PC's case, originally was called **scan line interleaved (SLI)** because every other scan line making up a screen display was divided between two graphics cards. Nvidia now prefers the term **scalable link interface** because it has developed more than one way to employ the two linked cards.

**3** In each type of graphics system, the paired cards split the geometrical data that describes the game's three-dimensional world. Each card has a **geometry engine** that translates the mathematical description of a scene or a character into a specific set of points in the 3D arena. Each card's processor performs **rendering** or **rasterization**, which is crunching the intensive math needed to translate the data for a three-dimensional world into polygons that cover a two-dimensional screen. Many GPUs are programmable, meaning that software can teach them new tricks.

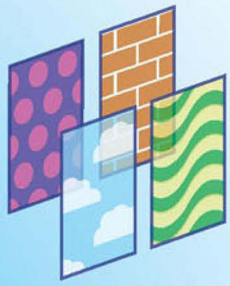
**4** Both graphics systems use **scissors rendering** to split the work load. One card takes responsibility for the top half of the screen, more or less, and the other card renders the bottom half. If one of the halves is simpler to render, such as a cloudless sky, that card will take on some of the other card's burden to equalize their loads.



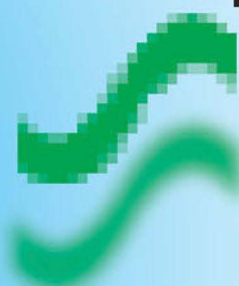
**5** ATI also provides **supertiling**. This rendering scheme breaks the work into a checkerboard pattern. One card takes the black squares and the other card works on the red squares. A chip on one of the boards shuffles the squares into the proper order before sending the instructions to the display.



**6** An on-chip feature called a **shader** has responsibilities far beyond those that it started with, which had been to paint the pixels of each polygon with the proper shading to make them appear three-dimensional. Today's shaders also supply textures, height, specific gravity, weight, and a host of other features so an object looks and reacts realistically according to laws of physics. The first time the shader pulls any of these properties from RAM, it stashes copies of them in faster memory chips on the video card. When the shader needs those same properties, it retrieves them faster from the onboard memory.



**7** The rendering engines always try to stay a step or two ahead of what's being displayed. When both cards finish rendering a screen, the engines stuff the pixel values into **frame buffers**, sections of high-speed memory on the card, where they wait until it's that frame's turn to show up onscreen for its 33 milliseconds of fame.



**8** The dual cards can also be used to create **double anti-aliased** graphics. Jaggies along what should be smooth lines are caused by having too little information to properly render the edge. The two cards attack the problem from two approaches and combine their anti-aliasing to achieve smoothing effects one card alone could not, at least not without incurring a performance hit.

**16x PCI-Express bus:** With a peak bandwidth of 2.1GB/sec, it makes the **accelerated graphic port (AGP)** at 266MB/sec suddenly seem like a fogey.

**Auxiliary power connector:** Some cards require this connector to supply the extra power that powerful video cards demand.

**Capacitors:** Any spikes or surges in electricity are caught by capacitors, which smooth out the lumps in the current.

**Onboard RAM:** The higher-end cards have as much as 640MB of memory to store bitmaps and give the math and rendering engines enough room to do their calculations.

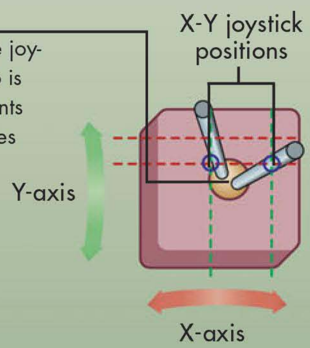
**Fan and heat sinks:** Needed on both the processor and memory chips to siphon off the extra heat the hard-working chips generate.

**Crystal:** Vibrates at a steady frequency to regulate all operations on the board. (The crystal may be susceptible to over-clocking.)

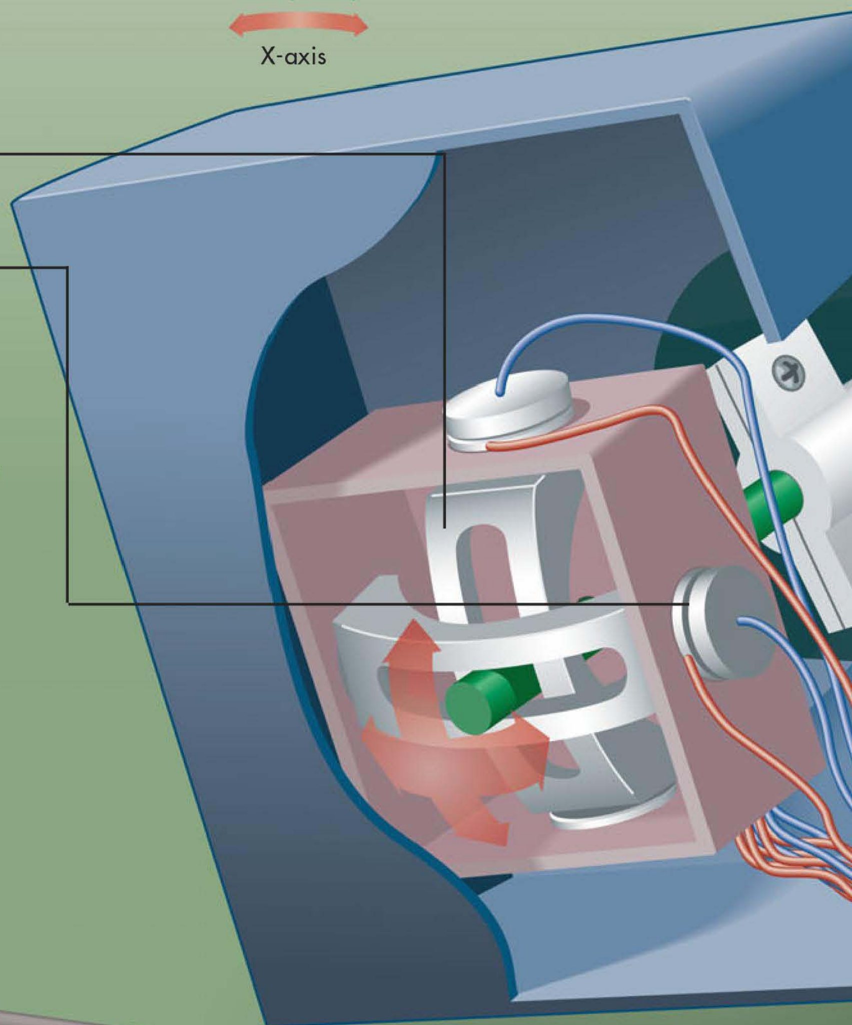
**TV tuner:** Not on all cards, but provides a way to display broadcast TV and/or to display the computer's own graphics on a TV screen.

# How a Joystick Puts You at the Controls

**1** All joysticks are designed to tell a computer how the handle of the joystick is positioned at any time. To do this, all the joystick has to do is provide the X-Y axis coordinates of the handle. The X-axis represents the side-to-side position. The path of the Y-axis is shifted 90 degrees from that of the X-axis and represents the forward-back position.

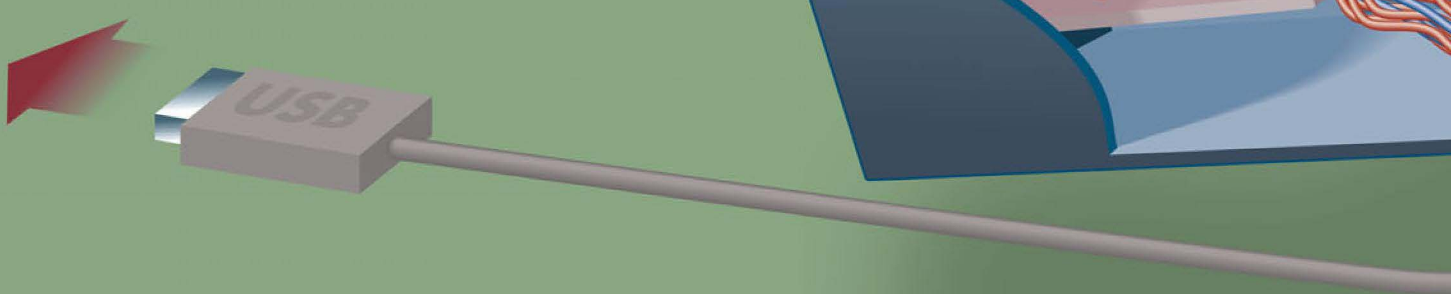



**2** The base of the handle is connected to a yoke with pivots that allow the joystick to move freely in any direction. Other types of game controllers, such as steering wheels and game pads, might look different from a joystick, but they produce signals using the same devices and connections a joystick does.



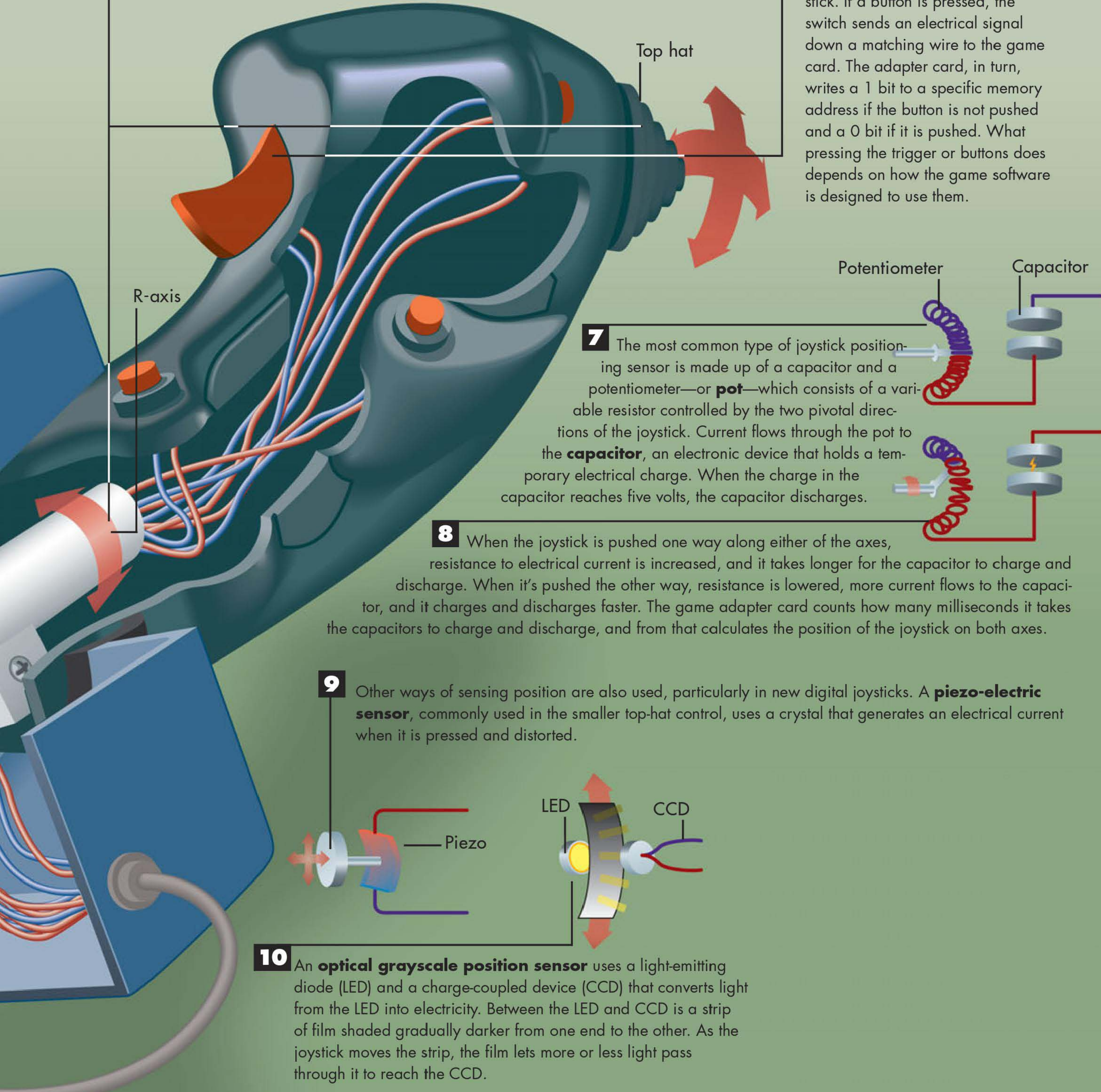
**3** Position sensors attached to each axis of the joystick respond to the joystick's X-Y coordinates and send signals to the game adapter card that the software uses to interpret the position of the game controller. The traditional joysticks create varying electrical analog signals through devices such as a **potentiometer**. Newer joysticks use a device that creates digital measures, often by registering how many and how quickly lights on the bottom of the joystick move past some point.

**4** Signals for the X-Y coordinates of the joystick are sent along a USB cable to the computer, where the software interprets them to figure out the joystick's position.



**5** More advanced joysticks use both sets of X-Y axis signals—one set for communicating the joystick position and the other to communicate the position of the **top hat**, an additional control that's maneuvered with the thumb. Some controllers also track the rotational movement (*R-axis*) of the joystick.

**6** Simple contact switches are used for the trigger and buttons on the joystick. If a button is pressed, the switch sends an electrical signal down a matching wire to the game card. The adapter card, in turn, writes a 1 bit to a specific memory address if the button is not pushed and a 0 bit if it is pushed. What pressing the trigger or buttons does depends on how the game software is designed to use them.




**7** The most common type of joystick positioning sensor is made up of a capacitor and a potentiometer—or **pot**—which consists of a variable resistor controlled by the two pivotal directions of the joystick. Current flows through the pot to the **capacitor**, an electronic device that holds a temporary electrical charge. When the charge in the capacitor reaches five volts, the capacitor discharges.

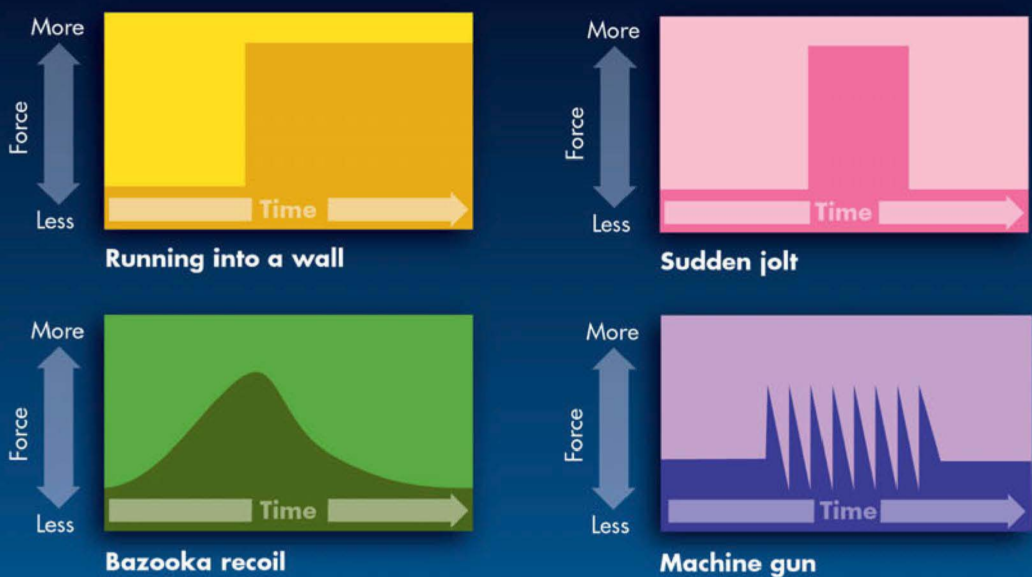
**8** When the joystick is pushed one way along either of the axes, resistance to electrical current is increased, and it takes longer for the capacitor to charge and discharge. When it's pushed the other way, resistance is lowered, more current flows to the capacitor, and it charges and discharges faster. The game adapter card counts how many milliseconds it takes the capacitors to charge and discharge, and from that calculates the position of the joystick on both axes.

**9** Other ways of sensing position are also used, particularly in new digital joysticks. A **piezo-electric sensor**, commonly used in the smaller top-hat control, uses a crystal that generates an electrical current when it is pressed and distorted.

**10** An **optical grayscale position sensor** uses a light-emitting diode (LED) and a charge-coupled device (CCD) that converts light from the LED into electricity. Between the LED and CCD is a strip of film shaded gradually darker from one end to the other. As the joystick moves the strip, the film lets more or less light pass through it to reach the CCD.

# How Force-Feedback Joysticks Work

**1** A **force-feedback joystick** responds to events in a game by moving the joystick on its own. The instructions to make a joystick move are created in software as wave forms, varying degrees of force graphed over periods of time. Here are examples of waveforms of different feedback forces.



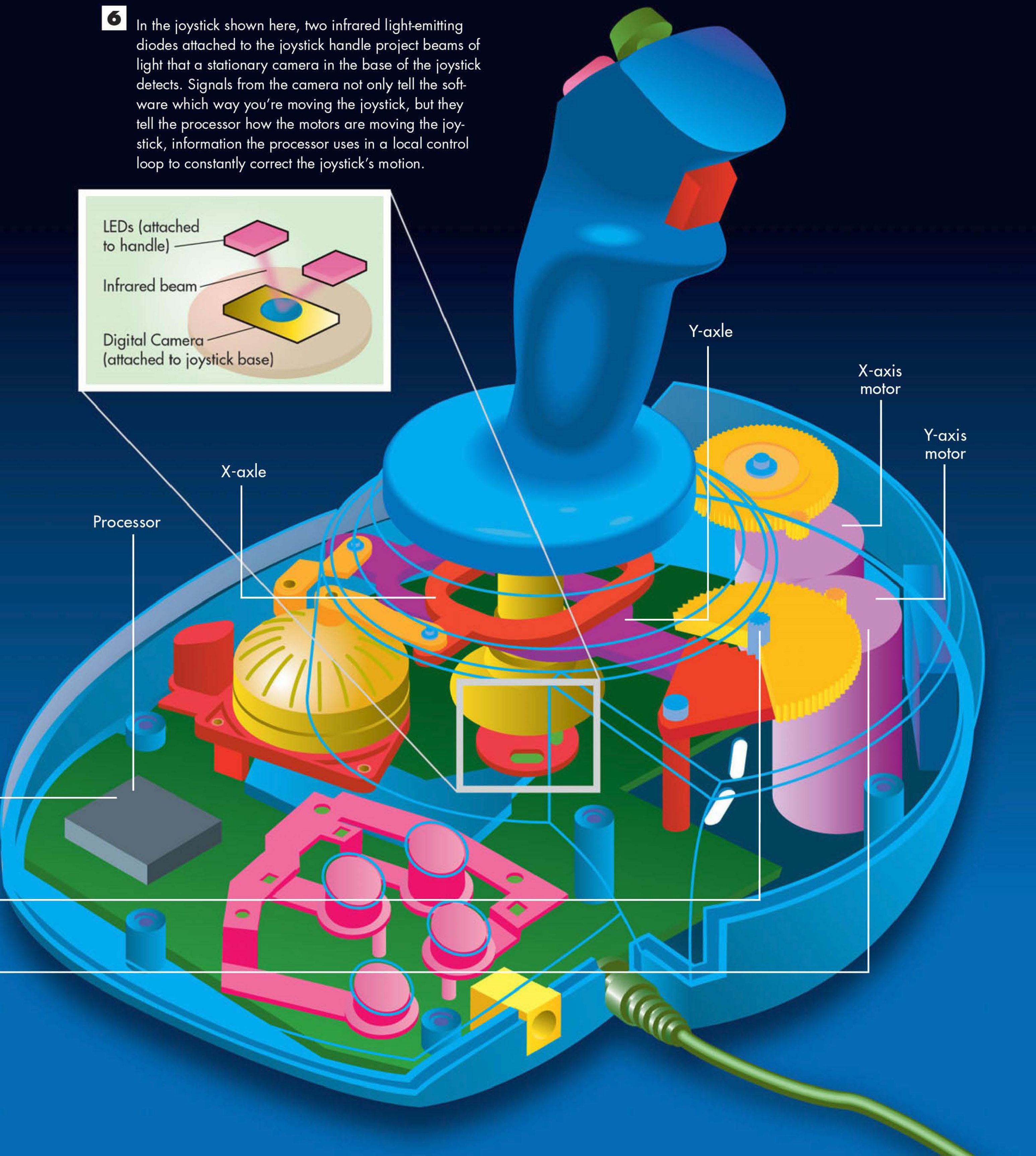

**2** The game software doesn't have to send the complete instructions for movements to the joystick. Instead, the software sends a **token**, a much shorter burst of data that identifies the complete wave form the joystick should execute.

**3** In the old Microsoft joystick shown here, a 16-bit, 25MHz microprocessor receives the token and looks up the wave form among 32 movement effects permanently stored on a ROM chip. The software can also download its own original wave forms to a 2KB RAM chip for the processor to use.

**4** The processor follows the instruction of the wave form that the token indicates and sends electronic signals to two motors—one each for the joystick's X-axis and Y-axis.

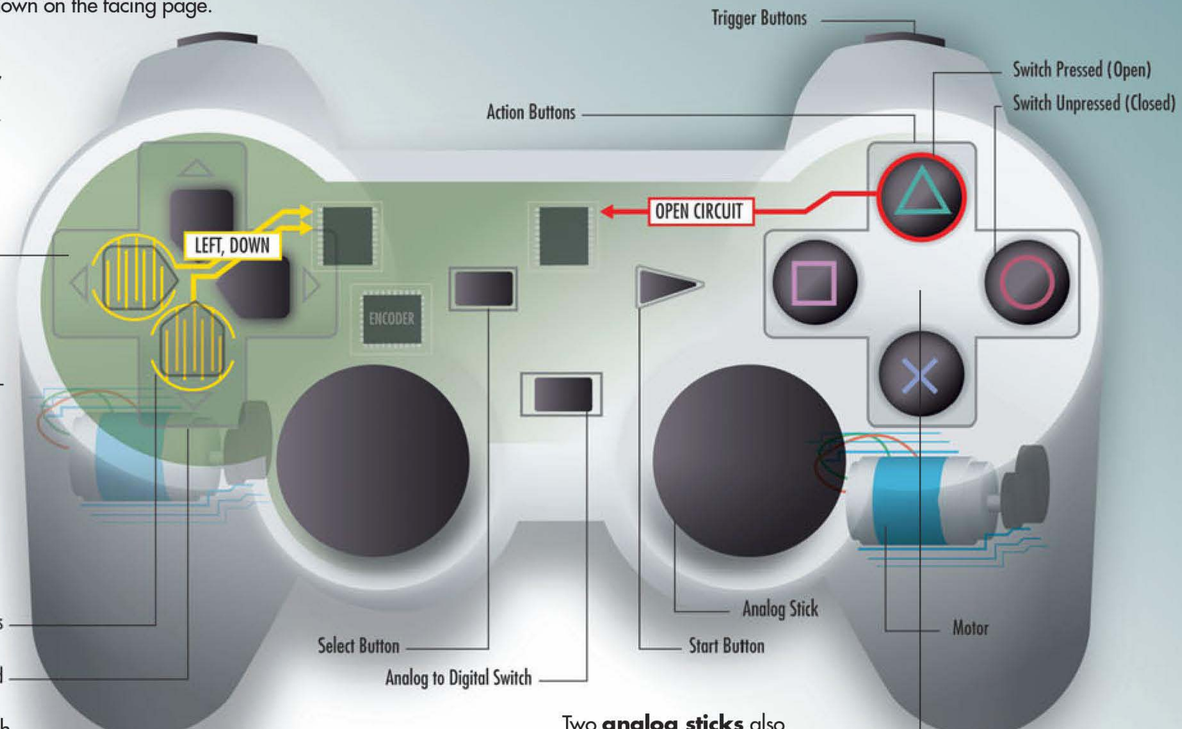
**5** The motors convey their precise movements to a gear train that transmits the forces to the joystick's two axes, causing the joystick itself to exert pressure.

**6** In the joystick shown here, two infrared light-emitting diodes attached to the joystick handle project beams of light that a stationary camera in the base of the joystick detects. Signals from the camera not only tell the software which way you're moving the joystick, but they tell the processor how the motors are moving the joystick, information the processor uses in a local control loop to constantly correct the joystick's motion.



# How Game Controllers Put Play at Your Fingertips

For years, the term *game controller* was synonymous with *joystick*. Even *Flight Simulator*, for many years the most elaborate and realistic of games, needed little more than a joystick, a trigger, and a button or two. But as games grew more versatile, players needed more versatile tools to control the action. From that need and from hand-held players such as the GameBoy came today's most prevalent controller—a device studded with buttons and pads, and held with two hands, as if at any moment it might try to escape. The Playstation's Dual Shock, shown here, is typical of most gamepads. Despite their almost universal use, little is intuitive about using them,a fact that has inspired the next generation of digital toys, including the Wiimote, shown on the facing page.




The **D-pad**, or directional pad, is one of the controllers' substitutes for a joystick. The thumb pressing on it can activate two **potentiometers** at the same time. Potentiometers consist of two current conducting strips separated by a material that resists current. The harder the D-pad presses against the potentiometer, the closer the two conducting strips get to each other, allowing more current to pass through them. A microchip combines the strength of the two signals to calculate a direction represented by the pad's use.

## The Gamepad

Four **trigger buttons** are located on the front of the controller. They may be used for shooting actions or in combination with other keys for more complex actions.

A switch changes the operation of the action and trigger button between **analog and digital signals.** When they are in analog mode, the buttons register both how long a button is pressed and the amount of pressure put on them. They measure pressure on a scale of 256 different amounts of pressure.

The Dual Shock has 14 switches, two motors, four analog controls, and a light. With signals coming and going to and from all those components, it would take a heavy cable to communicate them to the game console if it weren't for an **encoder** chip. It translates all analog signals to digital, compresses the data, adds tags that identify the control generating it, and sends the signals serially to the console, where the software interprets the signals in the context of the game being played.

Two **analog sticks** also substitute for joysticks. The sticks register the direction in which pressure is applied, much like a conventional arcade joystick, often to control player movement within the game. One of the sticks is commonly use to take over the functions of the D-pad while the other may control the camera view shown in the game.

Two motors in the handles of the gamepad provide **force feedback** to accompany explosions, gun recoils, blows, and other events, accompanied in real life by vibration and movement. Each of the motors connects to off-center weights. When the motors spin, the unevenness of the weights generates unbalanced force feedback sensations.

Of the more than a dozen buttons on most gamepads, the four **action buttons** on the right side of the Dual Shock are the most used. Gamers push these buttons in arcane patterns to make their game characters somersault, duck, fend off attacks, and decapitate opponents. Unlike most switches, which allow current through when they're pressed, gamepad switches are normally closed, allowing a small current to pass through until they are pressed, cutting off the electricity. The constant current tells the microchip overseeing the signals that the gamepad and switches are working; they aren't open because of a malfunction.

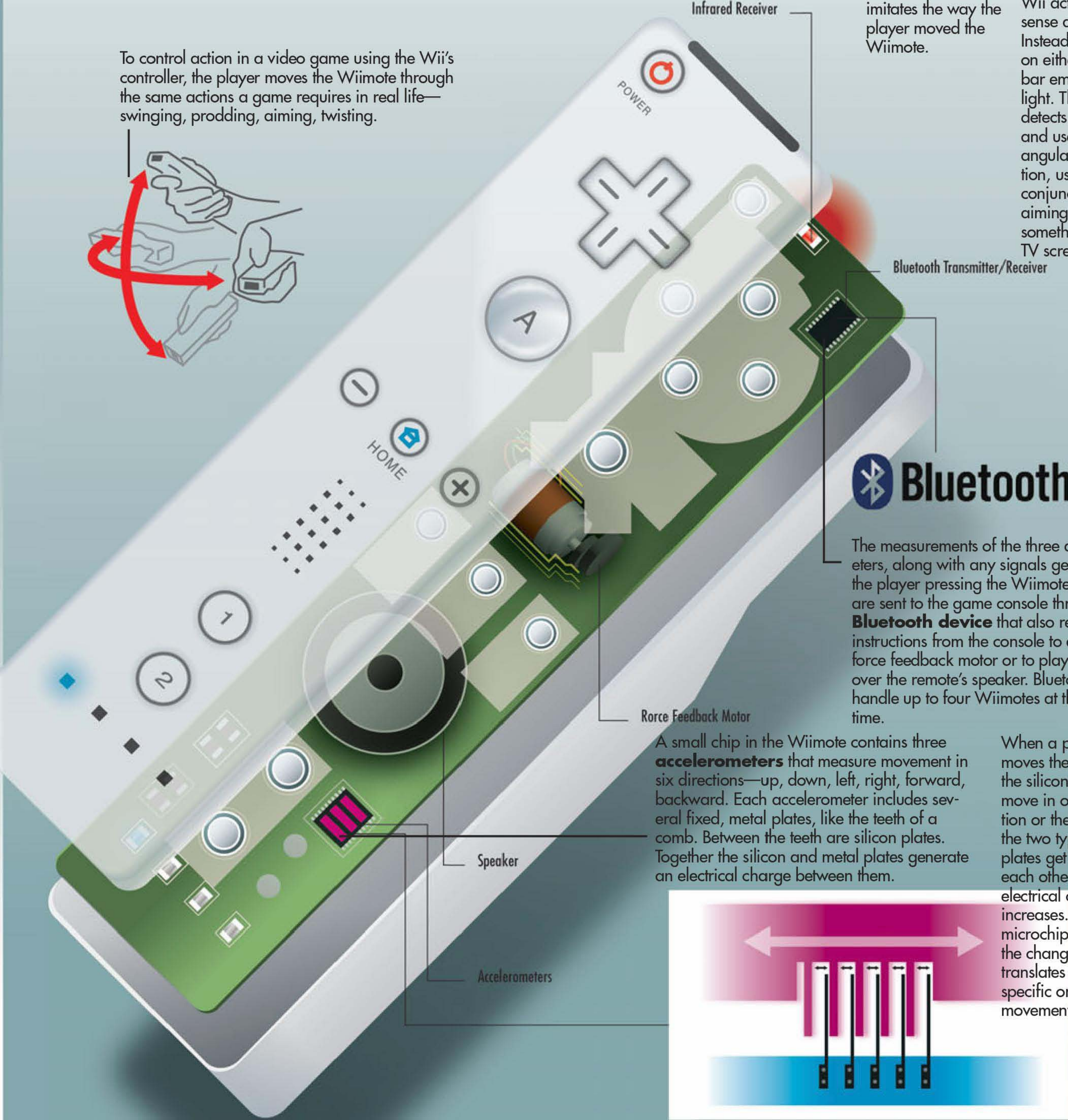
# The Wiimote

With most games, the way gamepads are used has little similarity to what happens onscreen. Delivering a kick to an opponent may be accomplished by pressing the action button sequence Left, Left, Up, Down. It's like talking in Morse code. That changed in 2005 when Nintendo showed the controller for its forthcoming Wii game system. It had no joystick, analog stick, or D-pad and only a few buttons. It looked more like a TV remote control than a game controller. In fact, Nintendo calls it the Remote; everyone else calls it a Wiimote. It also came without obscure button sequences to remember. Its versatility is in its simplicity. Swing it, and it's a bat, a golf club, or a tennis racket. Point it and push a button and it's a gun. Hold it in two hands and it's a steering wheel. To play, you do with the Wiimote what would be natural with a bat, a gun, or a steering wheel. It revolutionizes action video games.

The game's software combines the information sent to it through the bluetooth signals and translates that into onscreen action that imitates the way the player moved the Wiimote.

A **sensor bar** placed above or below the television connected to the Wii actually doesn't sense anything. Instead, five lights on either end of the bar emit infrared light. The Wiimote detects the lights and uses them to triangulate its position, usually in conjunction with aiming and firing at something on the TV screen.

To control action in a video game using the Wii's controller, the player moves the Wiimote through the same actions a game requires in real life—swinging, prodding, aiming, twisting.



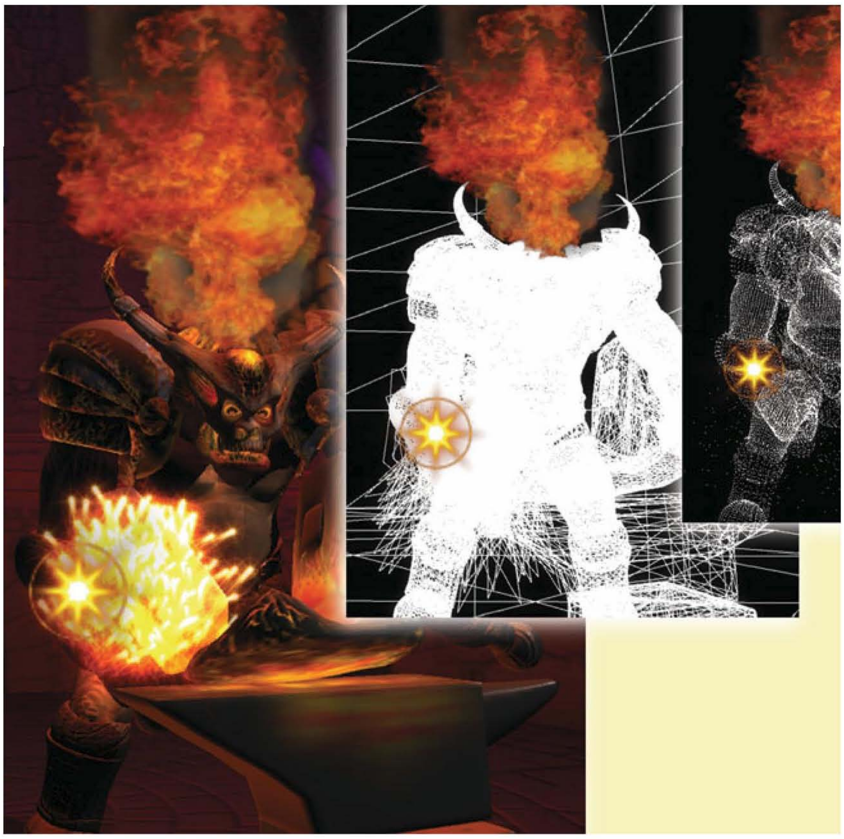

The measurements of the three accelerometers, along with any signals generated by the player pressing the Wiimote's button, are sent to the game console through a **Bluetooth device** that also receives instructions from the console to activate the force feedback motor or to play a sound over the remote's speaker. Bluetooth can handle up to four Wiimotes at the same time.

A small chip in the Wiimote contains three **accelerometers** that measure movement in six directions—up, down, left, right, forward, backward. Each accelerometer includes several fixed, metal plates, like the teeth of a comb. Between the teeth are silicon plates. Together the silicon and metal plates generate an electrical charge between them.

When a player moves the remote, the silicon plates move in one direction or the other. As the two types of plates get closer to each other, their electrical charge increases. A microchip measures the change and translates that into a specific onscreen movement.

CHAPTER 23

# How Games Create 3D Worlds



**IF** most men using computers would admit it—and of course, they won't; men never admit anything they're not forced to—the real reason for buying that computer was not to do the taxes, work brought home from the office, and an inventory of all those valuable tools in the garage. The guy got it to play games.

When I was shopping for my first personal computer, some 25 years ago, it had to be able to run Microsoft's *Flight Simulator*. Looking back, *Flight Simulator* wasn't the most exciting experience you could have. Basically you controlled a plane simulation, taking off, flying around for a bit, and then landing—if you could; I never mastered it and ended my flights by crashing in as spectacular a manner as possible, usually by slamming it into the Sears Tower in Chicago. (It never once occurred to me back then that someday someone might use *Flight Simulator* in a similar way, but for much more sinister purposes.)

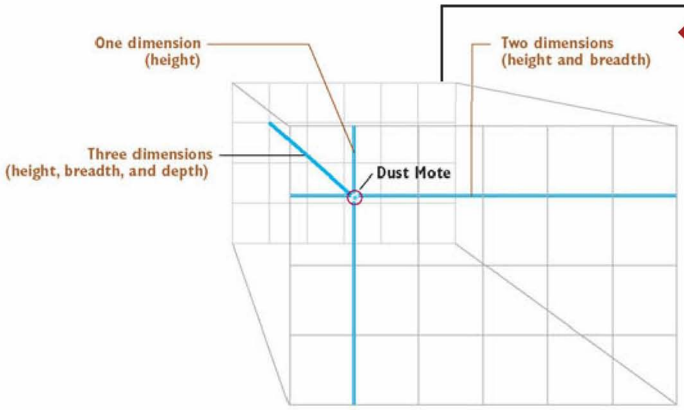
Back then, *Flight Simulator* was an exceptionally sophisticated piece of software; although unless you had propellers in your eyes, flying 300 miles in a straight line is, well, boring. Where are the obstacles to avoid, the enemy planes to battle, the drama, the heroism? It was boring for exactly the same reason it was such a hit among real pilots. Everything, from the layout of the plane console to the time it took you to fly anywhere was realistic. You can get realism in real life. We play games for a little unreality. The unreal has stirred brains and imaginations, dreams spawn inspiration, and the new and the different spur change and growth. Microsoft had likely reached similar conclusions when they eventual released *Combat Flight Simulator* in 1998.

And yet computer games get a bad rap, the same kind of bad-mouthing aimed at comic books, outer space movies, and Looney Toons when I was a kid. These were the sparks that ignited my visions. And yet I look back at those same sparks today and they seem so dull. The giant ants of *Them* are puny mindbogglers compared to the creatures of *Alien*. An episode of *Deadwood* has more complexity, characterization, irony, inspired use of language, and more challenges for my brain than most novels I read for English courses. Computer games—the good ones—immerse you in an alternative reality, force you to use your mind as well as your fire button, and stimulate you many times over what a good ol' game of checkers can do.

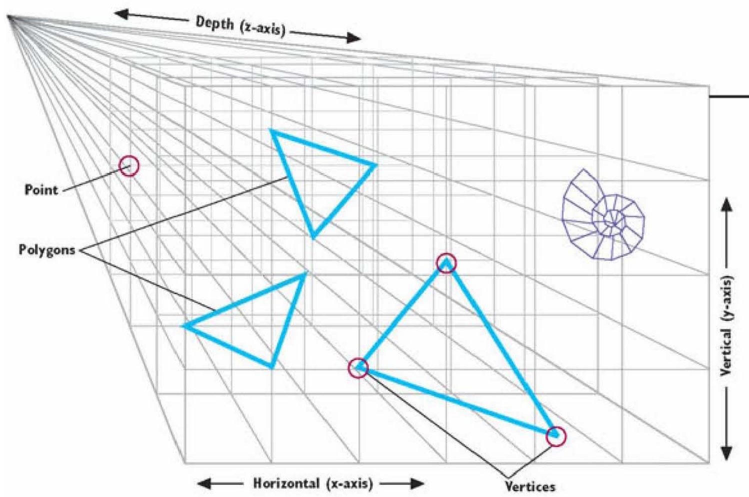
So don't keep your DVD of *Bioshock* at the office hidden in a drawer. Don't be ashamed because you sent the kids to bed early so you can play with the Xbox 360. You're exercising your brain, and if anyone doesn't understand that, you'd be happy to explain it to them during a deathmatch battle of *Unreal Tournament 3*.

# How Computers Plot a 3D World

## How Games Create 3D Worlds

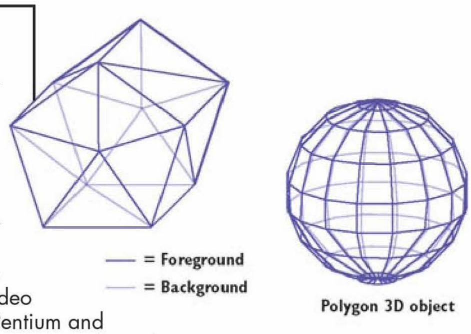



**1** Imagine a speck of dust floating near your head. As long as it stays put, you have no problem telling someone exactly where the speck is: Six and a half feet off the floor, 29 inches from the north wall, and a foot from the west wall. You need only three numbers and some agreed-upon starting points—the floor, the walls—to precisely pinpoint anything in the universe. (For our purposes, we'll ignore that curved space thing Einstein came up with.) That's how 3D games get started, by using three numbers to determine the position of all the important points in the graphic rendition of the world they're creating. Of course, today's PC games are pinpointing 47 billion dots a second, but the principle's the same as you putting numbers to the dust mote's location.



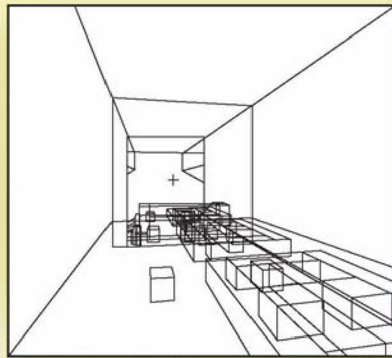

**2** In 3D games, as in life, we locate points along three axes: horizontal, vertical, and distal—the x-axis, y-axis, and z-axis, respectively. In three-dimensional space as well as games, three points are all that's needed to define a two-dimensional plane. 3D graphics create entire worlds and their populations from 2D **polygons**, usually triangles, because they have the fewest angles, or **vertices**, making them the easiest and quickest polygon to calculate. Most times, even a square, rectangle, or curve consists of combinations of flat triangles. (The vertices, as we'll soon see, are mere anchor points for straight lines.)

**3** Three-dimensional objects are created by connecting two-dimensional polygons. Even curved surfaces are made up of flat planes. The smaller the polygons, the more curved an object appears to be. The graphics processing unit on the video card (or cards) has a **geometry engine** that calculates the height, width, and depth information for each corner of every polygon in a 3D environment, a process called **tessellation**, or **triangulation**. The engine also figures out the current camera angle, or vantage point, which determines what part of a setting can be seen. For each frame, it rotates, resizes, or repositions the triangles as the viewpoint changes. Any *lines* outside the viewpoint are eliminated, or **clipped**. The engine also calculates the position of any **light sources** in relation to the polygons. Tessellation makes intense use of floating-point math. Without video cards with processors designed specifically for 3D graphics, the primary Pentium and Athlon CPUs in the computers would be woefully overtaxed. A changing scene must be redrawn at least 15 to 20 times a second for the eye to see smooth movement.

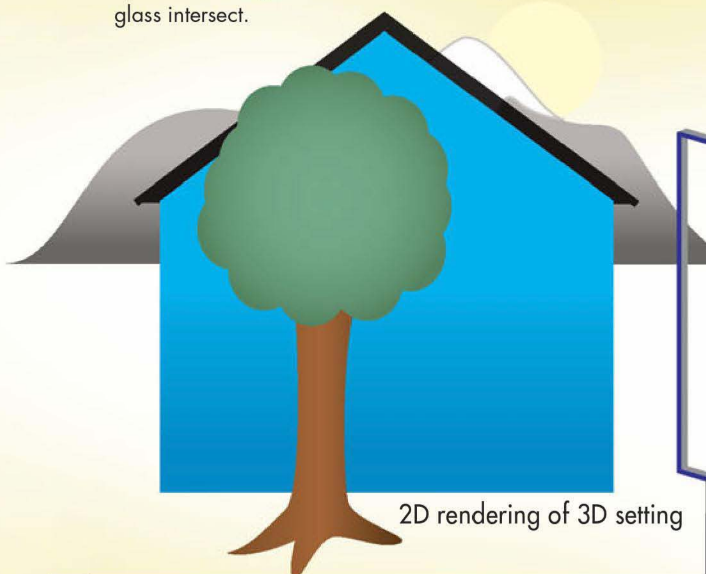


# How Geometry Becomes a Thing

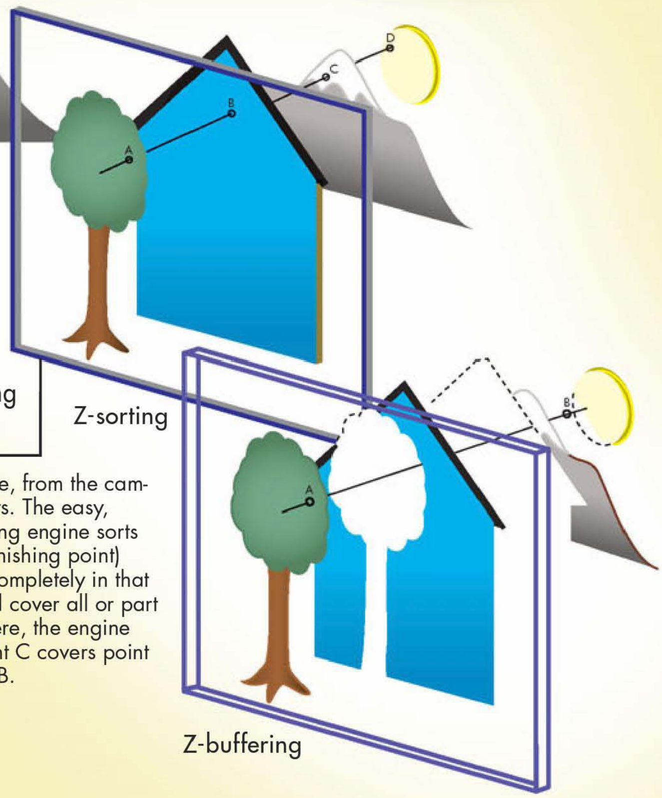
**1** The results of the geometry engine's calculations—the three-dimensional location of each vertex in the camera's viewing range—are passed to the **rendering engine**, another part of the **graphics processing unit (GPU)**. The rendering engine has the job of rasterization—figuring out a color value, pixel by pixel, for the entire 2D representation of the 3D scene. It first creates a wireframe view in which lines, or wires, connect all the vertices to define the polygons. The result is like looking at a world made of glass with lines visible only where the panes of glass intersect.



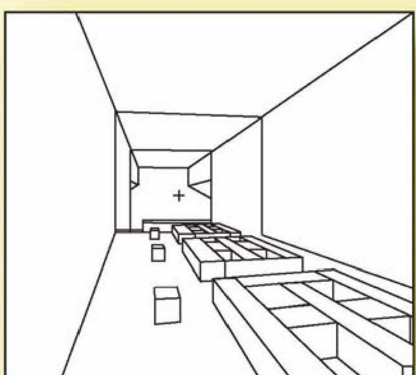
Wire frame view



2D rendering of 3D setting




**2** To create an illusion of depth, the 3D software must determine, from the camera's viewpoint, what objects are hidden behind other objects. The easy, memory-conserving way to do this is **z-sorting**. The rendering engine sorts each polygon from back (the objects closest to the distant vanishing point) along the z-axis to the front, and then draws each polygon completely in that order so objects nearer the vantage point are drawn last and cover all or part of the polygons behind them. In the illustration of z-sorting here, the engine renders all the points A, B, C, and D on the line AD. But point C covers point D, point B is painted over point C, and point A covers point B.



Hidden view

**3** **Z-buffering** is faster than z-sorting but requires more memory on the video card to record a depth value for each pixel that makes up the surface of all the polygons. Those pixels that are nearer the vantage point are given smaller values. Before a new pixel is drawn, its depth value is compared to that of the pixels along the same AB line that passes through all the layers of the image. A pixel is drawn only if its value is lower than that of all the other pixels along line AB. In the illustration here of z-buffering, pixel A is the only one the engine bothers to draw because pixels in the house, mountain, and sun along line AB would be covered by pixel A. With either z-sorting or z-buffering, the result is called a **hidden view** because it hides surfaces that should not be seen.

# How 3D Graphics Get Dressed

Drawing lines among all the vertices on a screen in the process of becoming a 3D image produces a wire frame, or **mesh**. But it's a naked mesh that, even in a hidden view, gives us only the rudimentary clues as to what all the meshes within the mesh depict. To turn these digital skeletons into digital objects is to clothe the mesh. The easiest technique to accomplish this is to run a flat layer of even color connecting all sides of each polygon, like a tent stretched taut over its poles. Needless to say, the effect is still more virtual cartoon than virtual reality. Few things, even when we try to make them so, are smooth and evenly colored. In the real world, objects have spots, streaks, grit, grain, bumps, lumps, and, in a word, texture. And so were born….

### Texture Maps

The early computerized 3D worlds tried to simulate realistic surfaces with **texture maps**. Texture maps are **bitmaps** (unchanging graphics) that cover surfaces like wallpaper—more specifically, like the bitmaps that can be titled to produce a seemingly seamless surface, a la Windows wallpaper. In this scene from a Quake level—and seen on the previous pages on wireframe and hidden views—texture maps are tiled to cover an entire surface and to simulate stone and lava. In simple 3D software, a distortion called **pixelation** occurs when the viewpoint moves close to a texture-mapped object: The details of the bitmap are enlarged and the surface looks as if it has large squares of color painted on it.

Texture maps


Distance fogged | Not fogged

### Fogging and Depth Cueing

Fogging and depth cueing are two sides of the same effect. **Fogging**, shown in the screen shot from *British Open Golf*—and not to be confused with the effect of wisps of fog that comes from **alpha blending**—combines distant areas of a scene with white to create a hazy horizon. **Depth cueing** adds black to colors by lowering the color value of distant objects so, for example, the end of a long hall is shrouded in darkness. The effect is not merely atmospheric. It relieves the rendering engine of the amount of detail it has to draw.

### MIP Mapping

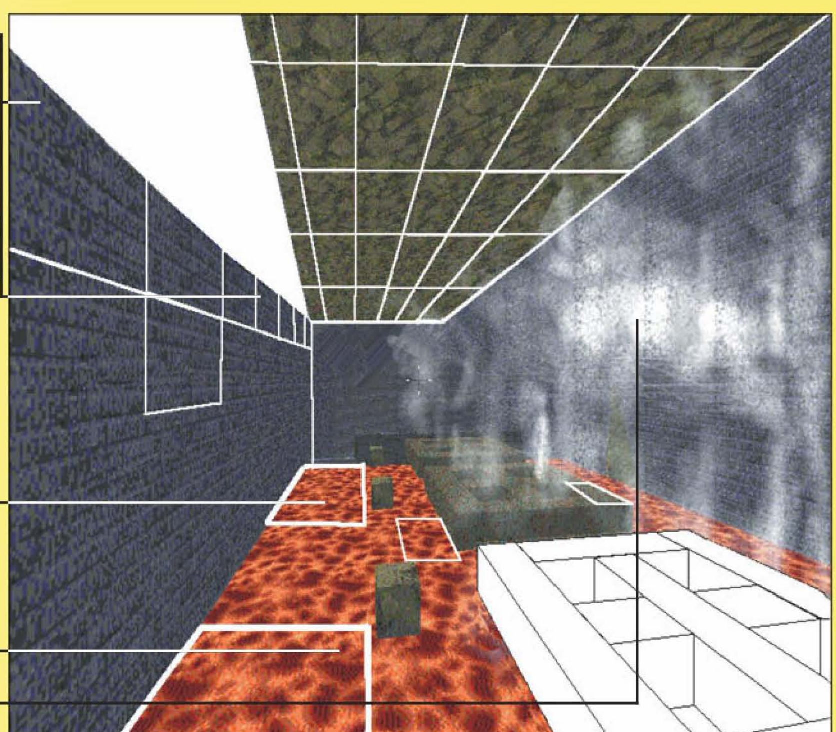
MIP mapping corrects for pixelation. The 3D application uses variations of the same texture map—MIP stands for *multim in parvum*, or "many in few"—at different resolutions, or sizes. One texture map is used if an object is close up, but a bitmap saved at a different resolution is applied when the same object is distant.

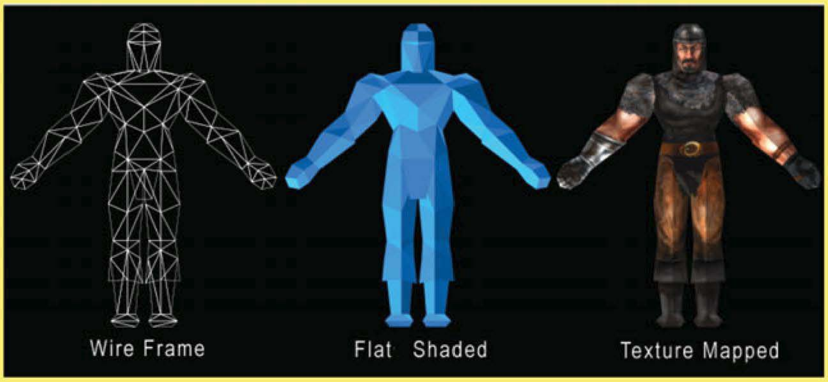
### Perspective Correction

Perspective correction makes tiles of texture maps at the far end of a wall narrower than tiles near the viewer and changes the shape of texture maps from rectangles to a wedge shape.

### Alpha Blending

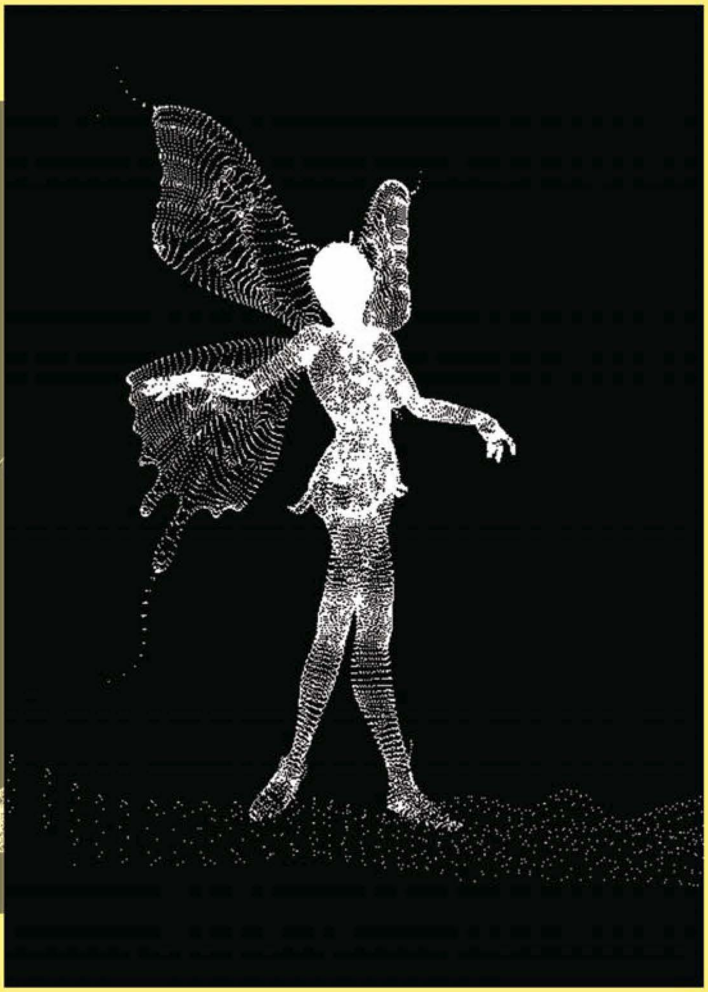
To create effects such as the semi-transparency that occurs through smoke or under water, the rendering engine uses **alpha blending** between texture maps that represent the surface of an object and other texture maps representing such transient conditions as fog, clouds, blurriness, or a spreading circle of light. The rendering engine compares the color of each **texel**—a texture map's equivalent of a pixel—in a texture map with a texel in the same location on a second bitmap. Then, it takes a percentage of each color and produces an alpha value somewhere between the two colors. A less memory-intensive way to accomplish a similar effect is **stippling** to blend two texture maps. Instead of performing calculations on each pair of texels, stippling simply draws the background texture and then overlays it with every other texel of the transparency texture.






## People Are Polygons, Too

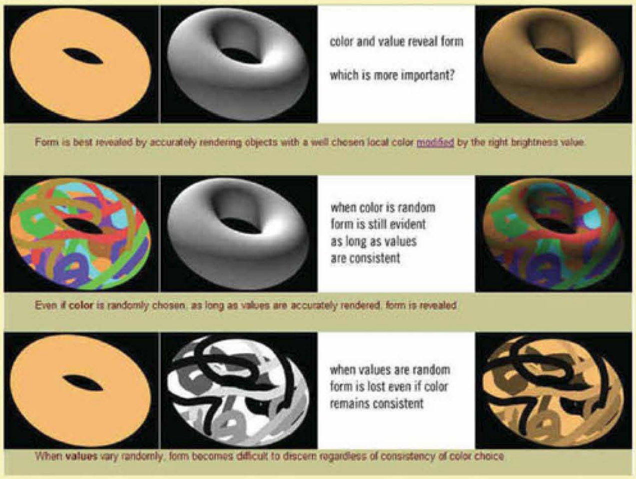
The techniques on the facing page were state of the art fewer years ago than this book has been in print. But they already look crude compared to the methods of rendering polygons today. In the screen shots here, Dawn, the fairy mascot of video chip maker nVidia, is a good example. Constructed entirely of polygons, she doesn't reveal her secret no matter how closely she's inspected. That's because she has so many polygons that when you see her in wireframe, as in the middle screenshot, the polygons blend into a white silhouette. It's only when you remove the lines that connect the vertices that you can see the construction of her body in the form of a few thousand points. A few years ago, it would have been beyond the capacity of PC video cards to construct such a creature in real time. Today's video cards, however, can construct and fill more than 6 million polygons every second.



# How Shaders Control the World

Vision includes **shape**, **color**, and, even more importantly, **value**, or **shading**. You don't have to have all of these to determine what you're looking at, but you do have to have shape and shading. Look at the depictions of the same object on the left. Color is not at all as helpful in identifying a shape as is the values—the intensities, or the lightness or darkness—of colors for different portions of an object.

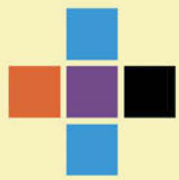
This is why 3D graphics gave birth to **shaders**. The first shaders did exactly what their name implies: They shaded certain polygons on specific sides of an object to give the illusion of depth and fullness. The technique began simply enough, using a single shade to each polygon, much as you might paint different walls of a room, but it became…

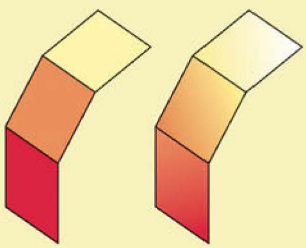



Joseph Francis, digitalartform.com

## Bilinear Filtering

Bilinear filtering smoothes the edges of textures by measuring the color values of four surrounding **texture-map pixels**, or **texels**, and then making the color value of the center texel an average of the four values. **Trilinear filtering** smoothes the transition from one MIP map to a different size of the same texture.





Flat shading

Gouraud shading

## Shading

Using information from the geometry engine about the location of light sources, the rendering engine applies shading to surfaces of the polygons. The rudimentary flat shading applies a single amount of light to an entire surface. Lighting changes only between one surface and an adjacent surface. A more sophisticated and realistic method is **Gouraud shading**, which takes the color values at each vertex of a polygon and interpolates a graduated shading extending along the surface of the polygon from each vertex to each of the other vertices.

## Ray Tracing

Before long, software and hardware developers realized they could have their shaders do more than apply different colors to polygons. They melded the abilities of shaders with other rendering techniques, such as **ray tracing**, which plots the path of rays of light as they are naturally reflected, absorbed, or refracted by the materials they shine on. The combination allowed shaders to create polygon chameleons who could take on qualities even real chameleons couldn't.

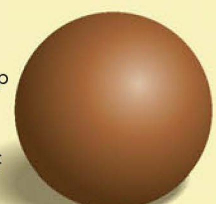
### Vertex Shaders

From Pacific Fighters, Ubisoft

The Gouraud shading technique has found new application in one of the most powerful rendering methods to date: **vertex shading**. Just as Gouraud shading creates a graduated shading stretching among the three vertices of a triangle, vertex shading does the same, only with any property animators want to assign an object, such as luminosity, temperature, and qualities that are not at all visual, such as weight and specific density. One of the most useful vertex shaders is **displacement**. In the screenshots above from the game Pacific Fighters, the one on the left is an ordinary animation with a necessarily limited number of polygons creating the ocean surface. On the right displacement, shaders control not just the height of the polygons but also the height of individual pixels making up the polygons for a more complex, more realistic surface.

From Vulcan, NVIDIA

### Particle Shaders



Vertex shaders are helpful in many areas where objects are so fluid or scattered that the traditional polygons handicap the realism of the animation. In these scenes from a short animation by NVIDIA featuring Vulcan, the god of fire, you can see how traditional polygons are used to construct Vulcan's body. But notice that the fire leaping from his body is not wireframed. That's because the animation uses **particle shaders**, which operate on each pixel independently of polygons to give a unified appearance to fluid objects that are nevertheless separate particles, or molecules.

# How Games Create New Worlds

Every day, all over the planet, thousands of people quietly disappear from our world and then materialize on worlds the rest of us have never seen. These worlds brim with monsters and wizardry, space aliens and people just trying to make a living. The escapees from our planet become fearless adventurers, plucky damsels, sorcerers' apprentices, or the bold leaders of hundreds of others who, too, had become bored with modern life, sat down at their computers, and transported themselves to alternate universes. They reappeared in a type of PC and console game called a **MMORPG** (**massively multi-player online role playing game**). The key word is *massively*. A successful MMORP has millions of paying users that are able to play the same game at the same time through their computers and the Internet.

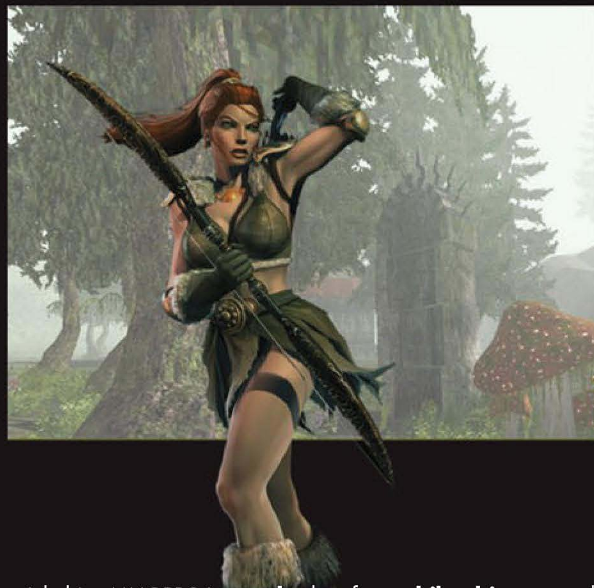
### Worlds Within Worlds

MMORPGs are a direct descendent of the non-computerized role-playing game *Dungeons and Dragons* (which also has an online game, called *Dungeons and Dragons Online*). That partially explains why so many MMORPGs have medieval settings, mythological monsters, and more than a modicum of magic. But MMORPGs are not limited to the classic D'n'D genre. You can also choose sci-fi MMORPGs, including *Star Wars Galaxies*, *EVE Online*, and *Tactical Commanders*. Back here on Earth, *Samurai Empire* puts you in a medieval Asian setting, whereas *City of Heroes* lets you become a full-fledged superhero.

**1** You take the first step toward fleeing this world when you install **client** software, bought off the shelf or downloaded, on your computer or game console. The client connects over the Internet to **server** programs that require scores of networked computers to hold all the information required to create a virtual world of the size you find in a MMORPG. One of the largest, *World of Warcraft*, covers more than 80 virtual square miles—four times the size of Manhattan. The servers must account for every blade of grass in the world—literally—every animal, the weather, changing seasons, and the characters not controlled by flesh-and-blood players.

**2** The first time you enter a MMORPG, you begin by creating your **avatar**, also called a **PC**, for **player character**. This is what other players will see when you're around them. You can change your hair color, bust size, even your gender or species. A game usually gives you a choice of characters that come with specialized skills, designed to work in concert with other player character types. A warrior, naturally, does well in battle, a wizard can cast damaging spells from a distance, while a healer might enhance or "buff" other characters while keeping them alive.

**3** If this is not your first time in the game, your client tells the server you want to join, and the server downloads to the client all the information about you that it needs to run the game on your end: your health, possessions, skills, and your appearance down to your clothing and hairstyle. The server and client do not keep in constant contact with each other. With thousands of players, reporting every little move would immediately clog the communication channels. The server generally maintains the overall status of the game, and the client is responsible for any local happenings, including the strenuous calculations necessary to present the 3D world on your PC's screen. At set intervals—called a **tick** in MMORPGs—the two exchange status updates and instructions.

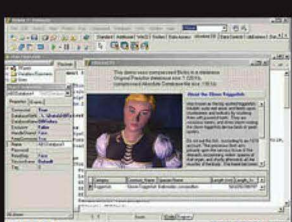


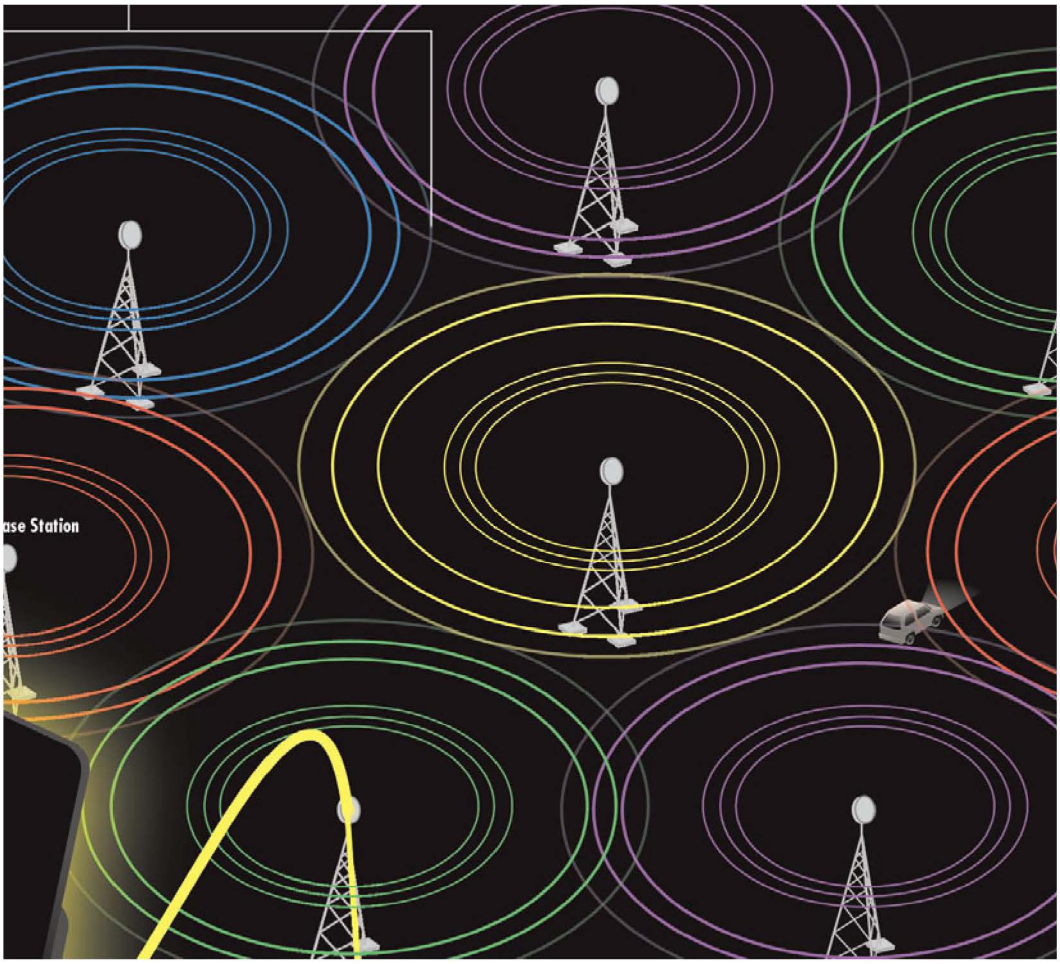
**4** When you step into the world, you encounter the avatars of other players. A computer server devoted solely to **chat** lets you communicate with other players. Some games have **player killers (PKs)** who prey on fellow playing characters, and some of the other PCs might look fearsome, but generally the games are designed so you gain more from cooperation than competition. Some games require the solving of puzzles, and if you're not a puzzle person, it's a good idea to travel with someone who is. And sometimes the only way to overcome some of the more dangerous creatures, such as dragons, is to have plenty of bodies to throw at them.

**5** The final type of creature to inhabit a MMORPG is a **mob**, short for **mobile object**, controlled by the game's servers rather than by players. Mobs include harmless woodland creatures and **non-player characters (NPCs)**, who are the cast of necessary bit players, such as merchants whose computer consciousness is limited to how much to charge for a loaf of bread or a new sword. (MMORPGs' economic systems spill over into the real world, where players on eBay sell MMORPG gold for actual greenbacks. Economists are studying the games as microcosms of **RL** (**real life**) economies.) Although merchants are technically mobs, the term is more identified with creatures more dangerous than salesmen, including evil magicians, dragons, orcs, giant sewer spiders, and zombies. The MMORPG constantly loses some of these creatures to the inevitable clashes with players, and the mobs must be regenerated. Certain spots in the world are often designated **spawning points**, or **camps**, where new bog slugs, swamp seeps, and gruttooth brutes come into the world.

**6** Although some MMORPGs, such as *Star Wars Galaxies*, let you develop a character who's a nonviolent artist or craftsman, fighting monster mobs is a large part of most multiplayer games. When you initiate a battle—by approaching a mob too closely or brandishing a weapon—the client reports the ensuing rumble to the server. The server notes what kind of mob it is, and looks up the creature's health and the attacks it can use. The server sends this information to the client, which executes the mob's attack using the mob's artificial intelligence.

**7** As you fight, the client gives the server an update of successful blows on both sides, and it deducts health points from both sides. When the number of points for you or your opponent falls below a certain level, one of you is dead. If it's not you, you and any other PCs who helped slay the beast are entitled to take from the mob whatever loot you want. The client depicts your avatar moving the loot from the mob; the server makes it official by moving the loot from the mob's data base record to yours. Killing mobs is also the only way you earn **experience**, the closest thing to a scoring system. With greater experience comes greater strength and endurance. Kind of ghoulish, but better than 8-to-5 at the office.





**900 B.C.** China has an organized postal service for government use.

**700 B.C.** Homing pigeons carry messages in ancient Greece.

**500 B.C.** Greek telegraph: trumpets, drums, shouting, beacon fires, smoke signals, mirrors.

**500 B.C.** Persia has a form of the Pony Express.

**200 B.C.** Tipao gazettes are circulated to Chinese officials.

**59 B.C.** Julius Caesar orders postings of *Acta Diurna*.

**100** Roman couriers carry government mail across the empire.

**1200** University of Paris starts messenger service.

**1200** European monasteries communicate by letter system.

**1533** A postmaster is appointed in England.

**1560** Legalized, regulated private postal systems grow in Europe.

**1609** First regularly published newspaper appears in Germany.

**1689** Newspapers are printed, at first as unfolded "broadsides."

**1785** Stagecoaches carry the mail between towns in the U.S.

**1819** Hans C. Oersted discovers that a wire carrying an electric current deflects a magnetic needle, a discovery that eventually leads to the creation of the telegraph.

**1839** John W. Draper and Samuel F. B. Morse photograph New Yorkers using a technique developed by Frenchman Louis Jacques Mande Daguerre. Draper and Morse are the first Americans to use the process.

P A R T

# 7

# How the Internet Works

C H A P T E R S



**1844**
The first public telegram is sent, using a system designed by Samuel Morse. Realizing the potential impact of this creation, Morse sends the message, "What hath God wrought!"

**1858**
The first transatlantic telegraph cable is completed and messages begin to flow between the shores of America and Europe. However, the cable fails after 26 days because the voltage is too high.

**1860**
On April 3 the Pony Express opens for business, pledging to "deliver the goods in 10 days or less." Its first route carries mail between St. Joseph, Missouri and San Francisco, California.

**1861**
Coast-to-coast telegraph communication begins in the United States.

**1861**
The last Pony Express run is made as the telegraph takes over.

**1866**
Transatlantic cable is finally completely successful. The cable remains in use for almost 100 years.

**1876**
March Alexander Graham Bell transmits the first message ever sent by telephone: "Mr. Watson, come here, I want you" to his assistant, who was linked by wire and receiver to the sending device in Bell's office.

**1877**
The first commercial telephone is introduced and the first telephone line is installed between Charlie William's electrical shop on Court Street in Boston and his home about three miles away.

**1886**
Heinrich Rudolf Hertz, of Megahertz (MHz) fame, proves that electricity is transmitted at the speed of light.

**1895**
Guglielmo Marconi sends a wireless signal using a directional antenna, prompting the development of the radio.

IT would be a lot easier to explain how the Internet works if you could hold it in your hand. Hardware—real, tangible, with a weight and size—is always easier to understand because you can see it and you can point with confidence to say this gizmo leads to that gadget, every time. The Net is not just a single thing; it is an abstract system. To understand the significance of this term, consider a less abstract system—your body.

The molecules that make up your body are not the same all your life. New molecules are constantly being taken in as food, water, and air, and are recombined into different molecules of muscle, blood, and bone. But no matter which molecules make up your hair and eyes and fingers, at any moment, the structure of your body remains the same. Your heart doesn't refuse to pump because new molecules of blood are created. If you remove some parts of your body, the system continues to function; sometimes, as in the case of brain damage, transferring the job of the missing parts to healthy parts of the brain.

As a system, the Internet is similar to a living organism. It grows, taking in new "molecules" in the form of PCs and networks that attach themselves to the Net. Parts of the Internet communicate with other parts that then respond with some action, not unlike the muscle activity set off by nerve impulses. You can think of the Internet as a network of networks. Amoeba-like smaller networks can break off the Net and live independent lives. Unlike amoebas, those smaller networks can later rejoin the main body of the Net.

The Net is ephemeral. Some pieces—the supercomputers that form the backbone of the Internet—are always there. The local area networks (LANs) found at countless businesses qualify as individual organs in the Body Internet. But nothing is really fixed in place—hard-wired. Each time you use your PC to connect with, say, a PC in Pittsburgh that maintains information on *Star Trek*, you don't have to use the same phone lines, switching devices, and intermediate networks to reach it. The route to Pittsburgh one time might run through Chicago; next time it might run through Copenhagen. Without realizing it, you can bounce back and forth among several networks from one end of the country to the other, across an ocean and back again, until you reach your destination in cyberspace.

**1901** Marconi sends a radio signal across the Atlantic.

**1902** Photoelectric scanning can send and receive a picture.

**1904** John Ambrose Fleming patents the first practical electron tube known as the Fleming Valve, based on Thomas Edison's patented Edison Effect.

**1909** In an event that will forever change the meaning of the word "news," a wireless telegraphic press message is sent.

**1915** AT&T researchers complete the first transcontinental call from New York to San Francisco and start experimentally transmitting voice across the country via radio.

**1920** KDKA of Pittsburgh begins operations by broadcasting the returns of the 1920 presidential election. Although fewer than 1,000 radios are tuned to this station, this is generally recognized as the beginning of commercial radio broadcasting in the United States.

**1923** "A picture in your radio set." In New York City, Russian-born engineer Vladimir Zworykin demonstrates his new invention, the iconoscope, which he claims will make it possible to transmit pictures—even moving pictures—through the air.

**1924** Pictures are transmitted over telephone lines.

**1925** AT&T's Long Lines Department offers the press an early facsimile service between New York, Chicago, and San Francisco.

**1934** The Communications Act of 1934 is passed. It is the first effort to regulate the telephone industry at the federal level.

It's a lot easier to say what you *can* get from the Net—information of *all* kinds. Being a system *without* physical limitations, it's theoretically possible for the Net to include all information on all computers everywhere, which in this age means essentially everything the human race knows, or thinks it knows. But because the Net is such an ad-hoc system, exploring it can be a challenge. And you don't always find exactly what you want. Plenty of software tools make surfing the Net easier, but the Internet itself has no overall design to help those using it. You're pretty much on your own when you jump in with whatever software you can find.

**1956 subv**

A sample of the cable laid in 1956 by the C. S. Monarch, connecting the United Kingdom and Newfoundland. The vacuum tube was part of several repeaters that periodically boosted the signal on its transatlantic trip.

***Courtesy of Lucent Technologies***

Despite the amorphous nature of the individual elements that make up the Internet, it is possible to describe the structure of the Net—the system that always remains the same even as the elements that make it up are changing from moment to moment. And for that structure, we can thank Sputnik.

## The Little Net That Grew

It was 1957 and the Cold War was subzero. The Soviet Union launched the first satellite, Sputnik, shaking the confidence of the United States in its scientific and technology leadership. President Eisenhower created the Advanced Research Projects Agency (ARPA) within the Department of Defense (DoD) to get the U.S. into space. That role was replaced by NASA, and ARPA redefined itself as a sponsor of advanced research projects at various universities and contractors.

In late 1960, Paul Baran of the RAND Corporation wrote a series of technical papers for the Pentagon analyzing the vulnerability of communications in case of a Soviet nuclear attack. A part of the papers were two ideas that would have far greater impact than anyone imagined. Baran said that military command messages and control signals should be carried over a distributed network that has redundant connections in

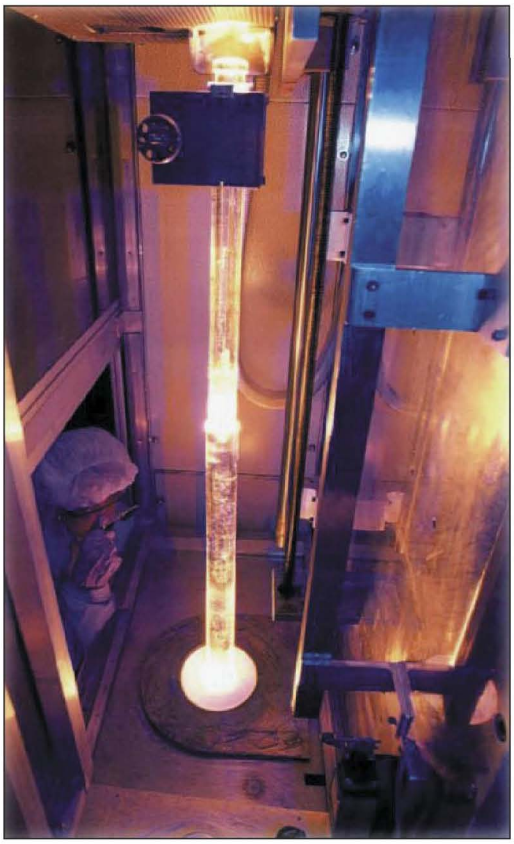
**1958**
The National Aeronautics and Space Administration (NASA) is enacted into law. Space programs and advanced strategic missile research are transferred to NASA. The ARPA budget is slashed to $150 million.

**1959**
ARPA redefines its charter as a "high-risk, high-gain" research sponsor, and more closely aligns itself with advanced research projects being conducted at various universities scattered around the U.S.

**1962**
The Internet is first conceived. Under the leadership of the Department of Defense's Advanced Research Project Agency (DARPA), it grows from a paper architecture into a small network (ARPAnet) intended to promote the sharing of supercomputers among researchers in the United States.

**1958**
In an effort to jump-start strategic missile research projects, President Dwight D. Eisenhower creates the Advanced Research Projects Agency (ARPA) within the Department of Defense (DoD). ARPA is given direction over all U.S. space programs and advanced strategic missile research.

**1960**
In a series of papers written for the Pentagon about the vulnerability of the military command and control system to nuclear attack, Paul Baran of the RAND Corporation introduces two revolutionary ideas that define a packet-switched network: Command and control messages should be carried on a distributed network with redundant connections to intelligent nodes, and each message should be broken into blocks and sent along a distributed network using a heuristic routing doctrine capable of routing itself along a damaged network.



**Optic Fiber**

Glass rods are placed in a draw tower and then super-heated, allowing miles of hair-thin optic fiber to be drawn and spooled. The fiber allows light to carry far more information than electricity running over copper wires.

case a missile took out part of the system. The best way to do this would be to break each message into blocks and send each block separately over that network, avoiding any parts that aren't working.

A specific distributed military network was never built, but a few years later ARPA began looking for a way its members could distribute messages and data among themselves so they could take advantage of each other's research. They came up with the idea of a distributed network called ARPAnet, built around something called interface message processor (IMP), which connected computers at university research centers. IMP also incorporated a technology called TCP/IP, developed at the National Science Foundation. Standing for transmission control protocol/Internet protocol, TCP/IP calls for breaking up messages and data into packets to which addressing information, error correction code, and identification are added. The packets can all travel to their destination over the distributed network, and a computer on the other end checks for mistakes and pieces them together in the right order. In 1969, computers at universities all over the country were linked to ARPAnet. The first letter sent over the new network was an "L" sent from UCLA to the Stanford Research Institute.

The ARPAnet continued to expand until, in 1972, it connected 23 host sites. By 1975, one new installation was being added each month. Meanwhile, other types of networks, such as the Computer Science Network (CSNET), designed to be a less expensive version of ARPAnet, sprang up across the country. In 1974, researcher Bob Kahn and Vint Cerf came up with the idea of a "network of networks" that would let dissimilar networks communicate with one another. By 1982, different networks were adopting TCP/IP as their communications standard, and the term "Internet" was used for the first time. A year later, a gateway was set up between CSNET and ARPAnet using TCP/IP as a common standard so people on the two networks could communicate with each other.

**1962**
AT&T places the first commercial communications satellite (Telstar I) into orbit.

**1962**
Joseph Licklider and Wesley Clark publish "On-Line Man Computer Communication," discussing their "Galactic Network" concept that would allow people to access data from any site connected through a vast network.

**1962**
Paul Baran of RAND develops the idea of distributed, packet-switching networks.

**1964**
The first local area network (LAN) is developed at Lawrence Livermore Labs.

**1965**
Thomas Merrill and Lawrence Roberts set up the first WAN (wide area network) between MIT's Lincoln Lab TX-2 and System Development Corporation's Q-32 in California.

**1967**
At a meeting of ARPA members, Larry Roberts, Network Project Manager, presents the concept of an ARPA network to connect and share research among its various sites.

**1969**
ARPANET IMP #2 is installed at the Stanford Research Institute, Menlo Park, California, and connected to SRI's SDS-940 Timesharing system and IMP #1. Several days later, Charley Kline, an undergraduate student at UCLA, becomes the first user of the ARPAnet when he types the letter "L" into the Sigma-7 and it is received on the SRI system.

**1972**
BBN's Ray Tomlinson creates the first software allowing email to be sent between computers. Email quickly becomes the network's most popular application.

A high-speed (56Kbps) backbone was built by the NSF to connect five supercomputing centers. All the transmission time wasn't used, so the NSF agreed to let local networks connect to each other through the backbone. The Internet was born, even if people didn't realize it yet. For years, the Internet was the territory of colleges and defense contractors. As the Internet grew, many of the people who nursed it through its infancy were dismayed when, in 1991, the NSF lifted restrictions on the commercial use of the Net. The first time someone sent out advertising over the Internet—a practice destined to earn the name spam—reactions among Internet purists were angry and vocal. And equally futile. The Internet and the World Wide Web, a section of the Internet developed to lift it out of its text-only origins into the world of graphics, sound, and video, had lives of their own. The imp was out of the bottle, and today what started as a modest experiment in communicating is growing at a rate of 100–200 percent a year.

## KEY CONCEPTS

**ADSL** Asymmetric Digital Subscriber Line: Modems attached to twisted pair copper wiring that transmit from 1.5Mbps to 9Mbps downstream (to the subscriber) and from 16Kbps to 800Kbps upstream, depending on line distance.

**analog** A signal that can take on any value in a range.

**asymmetrical** Provides different data rates in the upstream and downstream directions, where upstream is the direction from the end-user to the network, and downstream is the direction from the network to the user.

**bandwidth** The capacity of a channel to carry information. Measured in hertz (kHz or MHz) for analog transmission media, and in bits per second (kbps or Mbps) for digital transmission media. Literally, the width of a band of electromagnetic frequencies being used to send data. Wider bandwidths can deliver more information at the same time or send a given amount of data faster.

**bridge** A device that connects a local area network (LAN) to another local area network that uses the same protocol—for example, Ethernet to Ethernet or token ring to token ring.

**broadband** A term for high-speed, high-capacity Internet and data connections.

**backbone** The highest speed Internet or network routes, off which branch regional and local networks that make up the body of the Internet.

**browser** A PC program that displays information from the Internet.

**channel** A transmission path between two points. Channel usually refers to a one-way path, but when paths in the two directions of transmission are always associated, the term channel can refer to this two-way path.

**client** A computer or software that depends on another computer—a server—for data, other programs, or the processing of data. Part of a client-server network.

**1973**
Bob Metcalfe at Xerox's Palo Alto Research Center (PARC) develops a means to manage communications between numerous computers connected to a high-speed conduit using a technique called carrier sense multiple access/collision detection (CSMA/CD). He calls this means of communicating Ethernet.

**1974**
Bob Kahn and Vint Cerf jointly author a paper that addresses the issue of joining dissimilar networks via a gateway to create a "network of networks"—the Internet.

**1976**
Queen Elizabeth goes online with the first royal email message.

**1978**
Vint Cerf, Steve Crocker, and Danny Cohen create a plan to separate TCP's routing functions into a separate protocol called the Internet protocol (IP). Error handling and datagram functions would remain a part of TCP.

**1979**
Kevin MacKenzie sends the first ever emoticon in a message to the MsgGroup. The first is -), meaning tongue-in-cheek.

**1982**
The first PC LAN is demonstrated at the National Computer Conference by Drew Major, Kyle Powell, and Dale Neibaur. Their software would eventually become Novell's NetWare.

**1983**
A cellular-phone network is created.

**1983**
The Internet becomes a reality when the ARPAnet is split into military and civilian sections.

**distributed network** A network in which crucial files are spread across several servers. This eases the demand that would be made on a single server while safeguarding data, because information is stored on multiple servers, usually overlapping so that all data is available in its entirety even if one or more of the servers crashes.

**domain** A group of computers on a network that are administered as a unit, usually by the same company or organization.

**downstream** Refers to "host to end-user" (receive, download) direction.

**DSL** Digital Subscriber Line (DSL) technology provides a dedicated digital circuit between a residence and a telephone company's central office, allowing high-speed data transportation over existing twisted copper telephone lines.

**dynamic IP addressing** An IP address is assigned to the customer for the current session or some other ISP-specified amount of time.

**email** Electronic mail sent within a network or over the Internet.

**firewall** A security device that controls access from the Internet to a local network.

**gateway** Hardware and software that link two networks that work differently, such as a Novell and a Windows NT network.

**GIF** File extension for *graphics interchange format*; a compressed, bitmapped graphics format often used on the Web for animated graphics.

**HTML** Hypertext Markup Language, the coding used to control the look of documents on the World Wide Web.

**http** Part of a URL that identifies the location as one that uses HTML.

**hub** A device where various computers on a network or networks *on the* Internet connect.

**Internet** A worldwide network with more than 100 million users that are linked for the exchange of data, news, conversation, and commerce. The Internet is decentralized; that is, no one person, organization, or country controls the Net or what is available through it.

**IP (Internet provider)** A computer system that provides access to the Internet. AOL, Concentric, and most phone companies are IPs. Also stands for *Internet Protocol*, a format for contents and addresses of packets of information sent over the Net. Part of TCP/IP.

**IP address** An identifier for a computer or device on a TCP/IP network. Networks using the TCP/IP protocol route messages based on the IP address of the destination. The format of an IP address is a 32-bit numeric address written as four numbers separated by periods. Each number can be zero to 255.

**link** Text or graphics on a Web page that lead you to other pages if you click on them.

**local area network (LAN)** A more or less self-contained network (that can connect to the Internet), usually in a single office or building.

**network interface card (NIC)** A expansion board that allows a PC to connect to a network. Most NICs are designed for a particular type of network.

**peer-to-peer** A network in which there is no central server. All PCs on the network are peers and can perform the duties of a host and client at the same time.

**POTS** Plain Old Telephone System; basic analog telephone service with no frills from digital technology.

**router** A device that routes data between networks using IP addressing. Routers provide firewall security.

**1985**
The National Science Foundation (NSF) creates a national, high-speed (56Kbps) "backbone" network (NSFNET) connecting five supercomputing centers, most notably NCSA. NSF agrees to democratize the Net by allowing local networks to interconnect to the "backbone" and thereby each other.

**1986**
Albert Gore (D-TN) introduces the S 2594 Supercomputer Network Study Act of 1986.

**1988**
The NSFNET backbone is upgraded to T1 (1.544Mbps) and handles more than 75 million packets a day.

**1988**
The Internet Worm is released by Robert Morris Jr., affecting about 6,000 of the 60,000 hosts on the Internet.

**1989**
The first gateways between private electronic mail carriers and the Internet are established. CompuServe is connected through Ohio State University and MCI is connected through the Corporation for National Research Initiative.

**1990**
A happy victim of its own unplanned, unexpected success, ARPAnet is decommissioned, leaving only the vast network-of-networks called the Internet.

**1990**
The number of hosts exceeds 300,000.

**1991**
The World Wide Web (WWW) is released by CERN.

**1991**
Linus Torvalds announces Linux version 0.02.

**1991**
The number of Internet hosts breaks 600,000. NSF lifts restrictions on the commercial use of the NSFNET backbone. The NSFNET backbone is upgraded to T3 (44.736Mbps) as traffic passes 1 trillion bytes and 10 billion packets per month.

**search engine** A program that searches documents located on the Internet for key words or phrases entered by a person browsing the Net. It returns a lists of sites, sometimes rated, related to the topic searched for. HotBot, Yahoo!, and Excite are examples of sites that provide search engines.

**server** Part of a network that supplies files and services to clients. A *file server* is dedicated to storing files, and a *print server* provides printing for many PCs. A *mail server* handles mail within a network and with the Internet.

**spam** Electronic junk mail: Solicitations, usually to buy something, that are sent in email to hundreds or thousands of Internet users.

**spiders** Programs used by search engines to prowl the Web looking for new or changed pages. When a spider finds something new, it sends the information back to the search engine so it can update its index of subject matter and pages.

**static IP address** An assigned IP address used to connect to the Internet. The IP address stays with the customer's computer.

**switch** A device that provides communication channels among end-users. A circuit switch provides dedicated paths.

**T1** A point-to-point digital communications circuit with 25 channels, each of which carries 64,000 bits a second. The channels may be used for data or digitized voice.

**TCP/IP** Transmission Control Protocol/Internet Protocol; actually a collection of methods used to connect servers on the Internet and to exchange data. TCP/IP is a universal standard for connecting to the Net.

**URL** Uniform Resource Locator. A Web address expressed in English that takes a browser directly to a specific web page.

**Usenet** The world's largest system of on-going, online discussions by people who constitute newsgroups.

**Website** A group of World Wide Web pages, including a home page with links that lead to other pages at that site or on other sites.

**Webmaster** The person who maintains a website.

**wide area network (WAN)** Wide area network; a single network that extends beyond the boundaries of one office or building.

**World Wide Web** A loose confederation of Internet servers that support documents formatted in a language called HTML (Hypertext Markup Language) that can include links to other servers, documents, graphics, audio, and video.

**worm** A virus that doesn't infect other programs. It makes copies of itself, and infects more computers, often through network connections or as email attachments. A worm doesn't attach itself to legitimate programs as most types of viruses do, but it can still change or destroy other files like other types of virus do.

**XML** Extensible Markup Language. An advanced form of HTML, in which data is defined rather than simply formatted. For example, a browser could identify which numbers on a web page are prices and which are quantifiers, and act on that information.

**1992**
The number of Internet hosts breaks 1 million.

**1992**
The term "surfing the net" is coined by Jean Armour Polly.

**1992**
The first audio and video broadcasts take place over a portion of the Internet known as the MBONE.

**1993**
The White House and United Nations come online.

**1994**
The first cyberbank, First Virtual, opens.

**1994**
The number of Internet hosts breaks 3 million.

**1994**
Network Solutions, Inc. reports that it is registering domain names at the rate of 2,000 per month.

**1995**
A team of programmers at Sun Microsystems release an Internet programming language called Java, which radically alters the way applications and information can be retrieved, displayed, and used over the Internet.

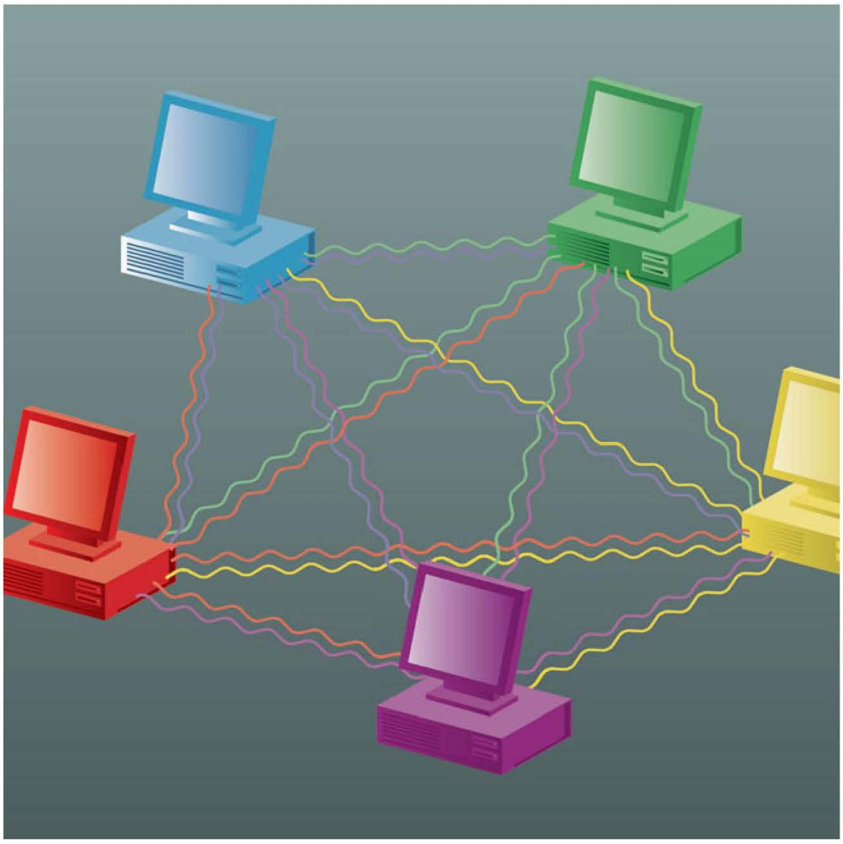
**1996**
As the Internet celebrates its 25th anniversary, the military strategies that influenced its birth become historical footnotes. Approximately 40 million people are connected to the Internet. More than $1 billion per year changes hands at Internet shopping malls, and Internet-related companies such as Netscape are the darlings of high-tech investors.

**1997**
In January, the number of Internet hosts breaks 16 million.

CHAPTER

24

# How Local Area Networks Work



A **local area network (LAN)** is, for many people, the entry point to the Internet. A LAN physically links several PCs to each other and often to a server that hosts shared data or provides access to the Internet. This is accomplished with a variety of materials—twisted-wire cables, fiber optics, phone lines, and even infrared light and radio signals.

Whatever the technology, the goal is the same—to send data from one place to another. Usually, the data is in the form of a message from one computer to another. The message might be a query for data, the reply to another PC's data request, an instruction to run a program that's stored on the network, or a message to be forwarded to the Internet.

If the data or program for which the message asks isn't on the Internet, it might be stored on a PC used by a co-worker on the network, or on a **file server**, which is a specialized computer. A file server is usually a high-performance PC with multiple large hard drives that are not used exclusively by any individual on the network. Instead, it exists only to serve all the other PCs using the network—called **clients**—by providing a common place to store data that can be retrieved as rapidly as possible by the clients. Similarly, a network might include an Internet server that links the LAN to the Net, CD-ROM jukebox servers, or print servers that everyone on the LAN can use for printing. A **print server** is a PC connected to a printer, or a network printer that can be connected to a network without an intervening PC.

If a network does not have a dedicated server, it is a **peer-to-peer network**. In a peer-to-peer network, each individual's PC acts as a server to other PCs—its peers—on the network and also is a client to all its peers acting as servers.

The network must receive requests for access to it from individual PCs, or **nodes**, linked to the network, and the network must have a way of handling simultaneous requests for its services. When a PC has the services of the network, the network needs a way of sending a message from one PC to another so that only the node for which it's intended recognizes it, and it doesn't pop up on some other unsuspecting PC. And the network must do all this as quickly as possible while spreading its services as evenly as possible among all the nodes on the LAN. LANs are a microcosm of the Internet, even as the LANs are a part of the Internet.

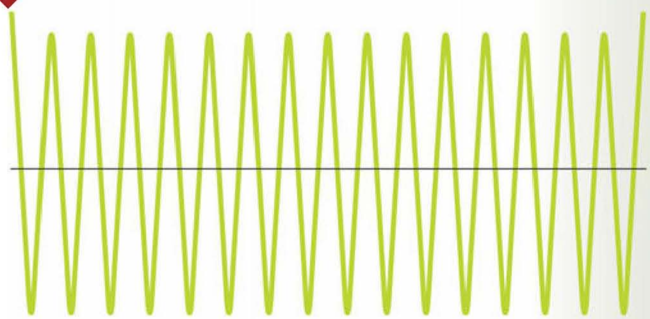
In this chapter, we'll look at the most common types of networks, including the notorious file-sharing networks and the works of the most common LAN configuration, Ethernet.

# How Packets Divvy Up Data

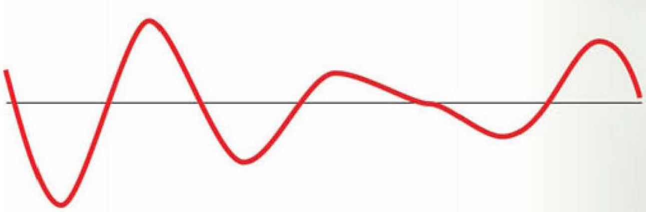
Sending information digitally isn't all that new. Samuel Morse sent the first telegraphed message in the U.S.—"A patient waiter is no loser."—in 1838. He used a binary system—dots and dashes—to represent letters in the alphabet. Before Morse, smoke signals did much the same thing, using small and large puffs of smoke from fires. But for a good chunk of the 20th century, analog signals in telephones, radio, recordings, and TV became the standard ways to send data over great distances. With networking and the Internet, however, digital communications are once more in vogue, even replacing analog signals used in television, radio, and telephone. What makes this all possible is something called a **packet**.

## How Analog Data Works

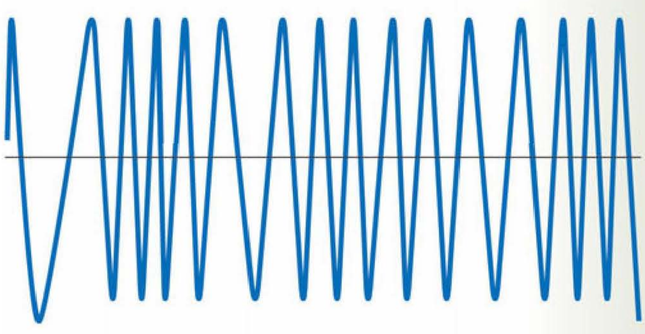
1 Analog communication works with two types of waves.



A **carrier wave**, or **carrier signal**, is a steady, strong wave that carries no information of its own. It is usually sinusoid, which means it has a constant waveform, both for **amplitude** (its loudness) and **frequency** (how many times it cycles from its high point to low point and back again in a second). An FM radio station broadcasting at 104.5 "on the dial!" is broadcasting a carrier that cycles 104.5 million times a second.

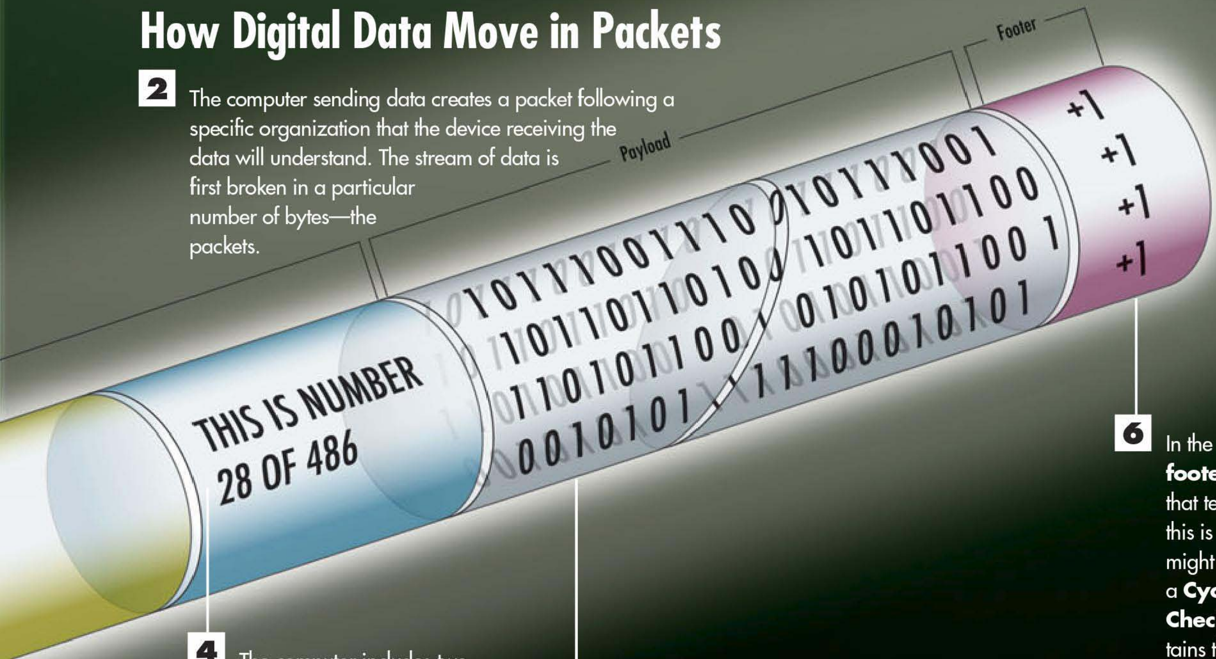
An **information wave** is produced by a microphone and a recording on tape, CD, or DVD. It lacks the carrier wave's strength to cover long distances. And unlike the carrier signal, it is irregular, changing form constantly due to the processes that produce it.



2 When the information wave is superimposed on the carrier, the information wave **modulates** the carrier (modulate simply means to change something). The informational signal could vary the amplitude or the frequency of the carrier. Speech is an example of a modulation. Vowels are the carrier that let you project your words far enough and clearly enough for others to hear you. Consonants are the information signals that transform the "ee" vowel into "me," "we," "see," and other variations on the basic "ee" sound. When the modulate signal is received (by a radio, TV, amplifier, or human ear), the receiver strips away the known values of the carrier signal. What is left is the original information signal.

**1** For digital data transfers, it's convenient to think of the **packet** as the equivalent of a carrier signal. A packet itself has no information, but it encloses, metaphorically at least, the real information traveling to another computer or to a component within the same computer. But there are more advantages in packets than simple bundling. Packets permit addressing, error correction, and the use of multiple pathways to get the information from one spot to another. The organization of data packets varies to match the type of data they contain, and they may be called **frames**, **segments**, or **blocks,** depending on their data. We'll look at the most common packet you'd encounter—the Internet packet.

# How Digital Data Move in Packets




**2** The computer sending data creates a packet following a specific organization that the device receiving the data will understand. The stream of data is first broken in a particular number of bytes—the packets.

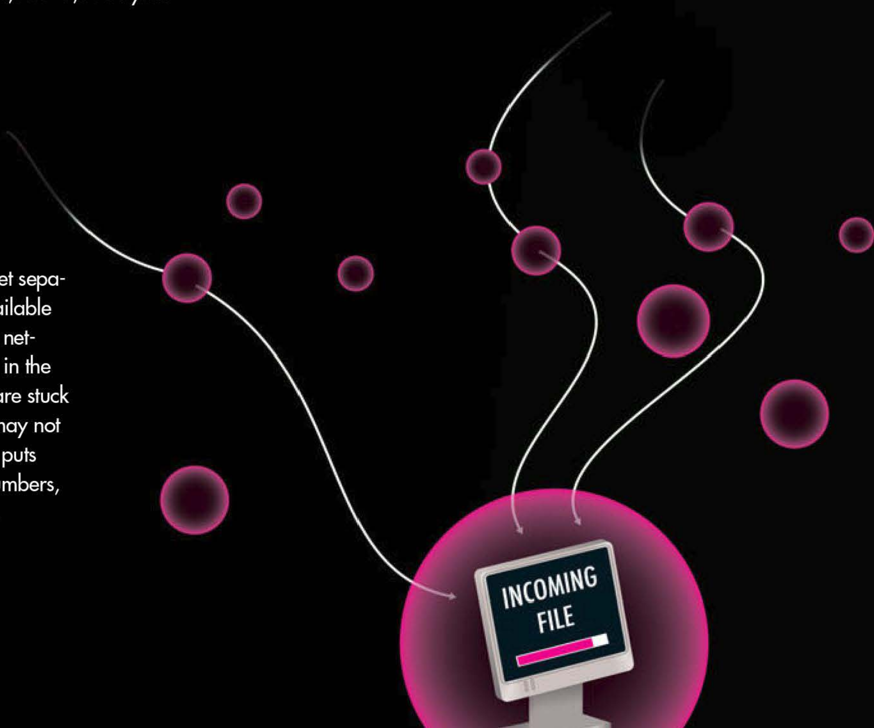
**4** The computer includes two numbers. The first is the number of packets the information is divided into. The second is the sequence number of this particular packet.

**5** The computer follows the form for bundling the actual data—the **payload**—set out by the **Transmission Control Protocol/Internet Protocol (TCP/IP).** Each packet holds 1,000–1,500 bytes.

**6** In the packet's **trailer** or **footer**, there are a few bits that tell the receiving computer this is the end of the packet. It might also include the results of a **Cyclic Redundancy Check (CRC)**. The CRC contains the sum of all the 1s in the packet. The receiving computer does the same calculation, and if the results don't match, the receiver asks the sender to retransmit the packet.

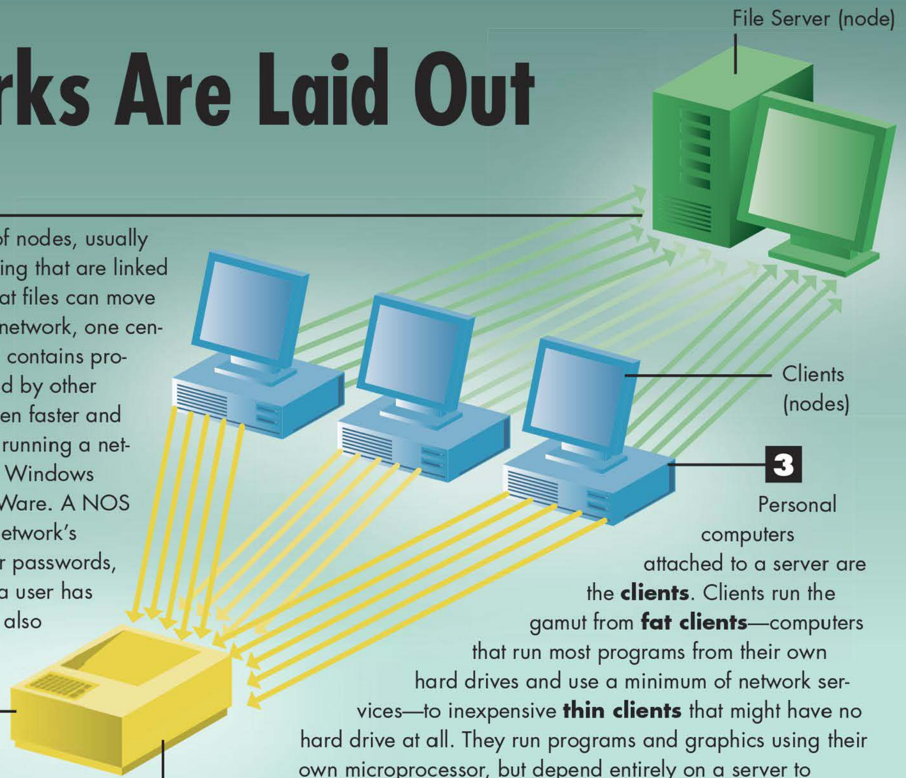
**3** To each packet's **header**, the computer adds the **IP address** that the packet is supposed to go to and the sender's IP address.

**7** The computer sends each packet into the Internet separately, and each packet takes the best route available at the time it shoves off. This method allows the network to spread the traffic out more evenly, and in the event of serious traffic jam, not all the packets are stuck in the same place. As the packets arrive, they may not be in the correct order. The receiving computer puts them into a buffer and, using their sequence numbers, builds the entire message as the packets arrive.



# How Networks Are Laid Out

## Client/Server Networks



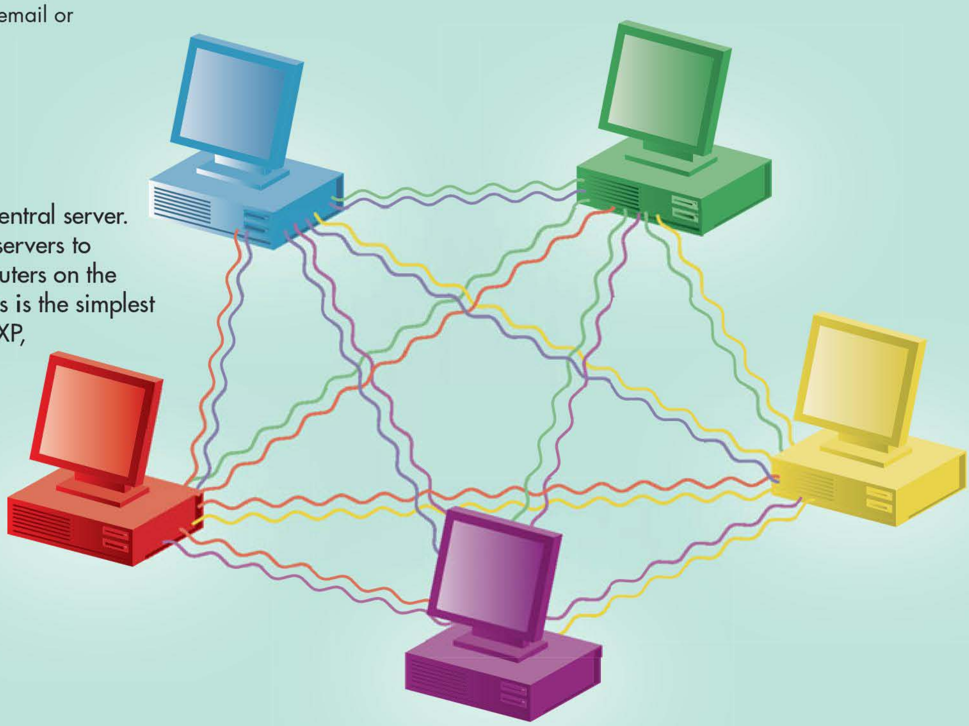

**1** A local area network (LAN) is made up of nodes, usually two or more computers in the same building that are linked together with wires or radio signals so that files can move among the computers. In a client/server network, one central computer is the file server. The server contains programs and data files that can be accessed by other computers in the network. Servers are often faster and more powerful than personal computers, running a network operating system, or NOS, such as Windows Server 2003, Unix, Linux, or Novell NetWare. A NOS manages the movement of files and the network's security by maintaining lists of users, their passwords, and the drives and directories for which a user has been given access privileges. A server is also called a host computer.

**2** Some servers specialize in functions other than passing out files. A print server lets everyone on a network share a printer. The printer can be attached to a computer on the network; some printers designed for network use can be connected directly to the network without a host PC. Other specialized servers provide shared access to the Internet, banks of CD-ROM drives, and tape backup. Some servers specialize in running programs that are designed for network-wide use, such as an email or database server.

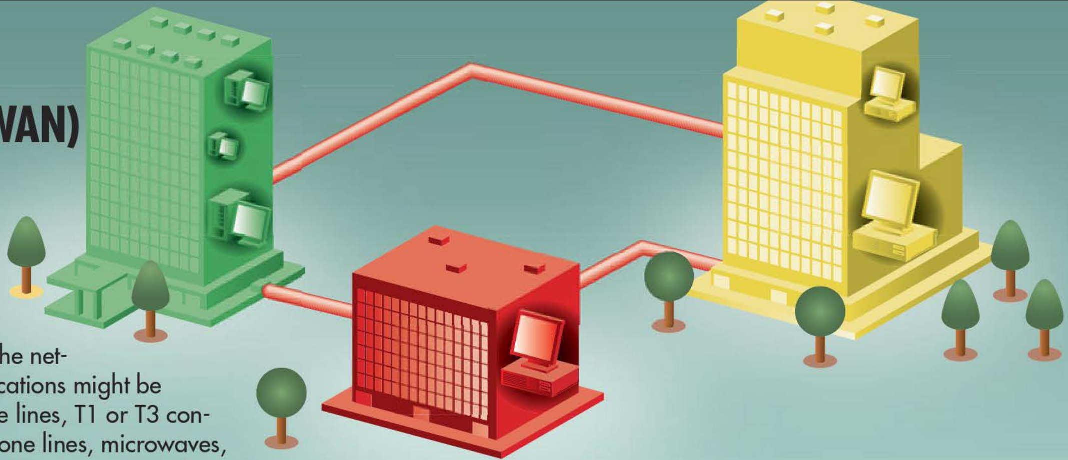
**3** Personal computers attached to a server are the **clients**. Clients run the gamut from **fat clients**—computers that run most programs from their own hard drives and use a minimum of network services—to inexpensive **thin clients** that might have no hard drive at all. They run programs and graphics using their own microprocessor, but depend entirely on a server to access programs and store data files. A **dumb terminal** is a monitor, keyboard, and the bare minimum of hardware needed to connect them to the network. It uses the server's microprocessor to perform all functions.

## Peer-to-Peer Networks

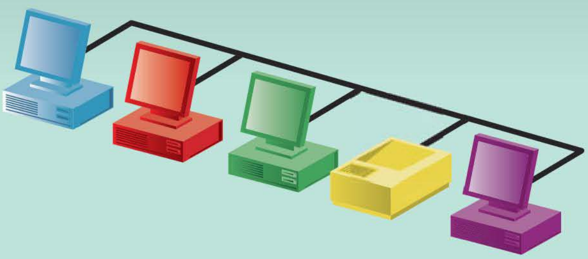
In a **peer-to-peer network**, there is no central server. Instead, all computers on the network act as servers to every other node. At the same time, all computers on the network act as clients to all the other PCs. This is the simplest type of network to install. Windows 98, Me, XP, and Vista come with the software to set up a peer-to-peer network.



## Wide-Area Network (WAN)



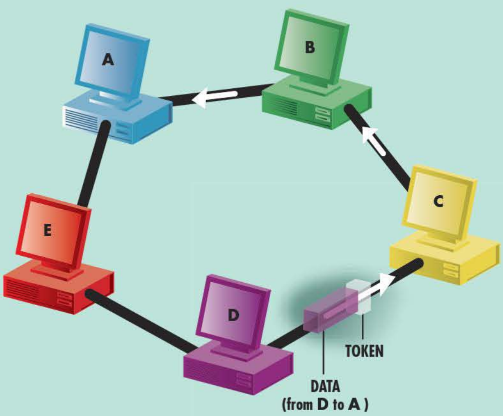
When components of a network are spread among several buildings, it becomes a **wide-area network**. Chunks of the network in different locations might be connected by phone lines, T1 or T3 connections, leased phone lines, microwaves, or the Internet itself. One way to use the Internet for a WAN is through a **virtual network**, software that uses heavy encryption to maintain privacy among Internet-connected PCs so that they work and fend off hackers as if the scattered nodes were at the ends of Ethernet cables in the next room.



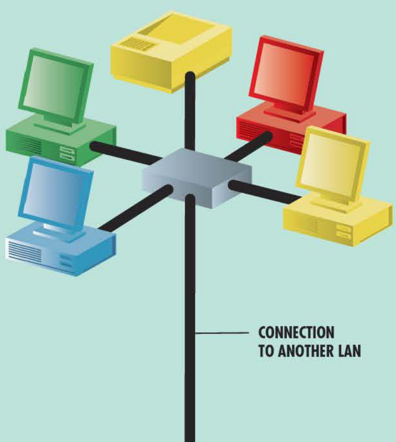
## The Shapes of LANs

The way that data moves from one node to others in a LAN determines the network's **topology**—its shape.

**Bus topology** All the nodes in a LAN are connected along a single cable—the bus—stretching from one node to the next. It is inexpensive and simple to set up, but a bad connection at one of the nodes also takes other nodes off the network.



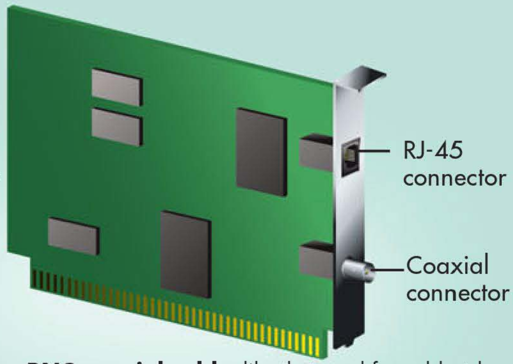

**Token ring** All the nodes are connected to a giant ring of cable that has no real beginning or end. Data travels from one node to another by one node grabbing a **token** of code that endlessly loops through the network. The node replaces the token on the ring with the node's data and the address of the node for which it's intended. The message circles through the ring until another node recognizes that the data is addressed to it.

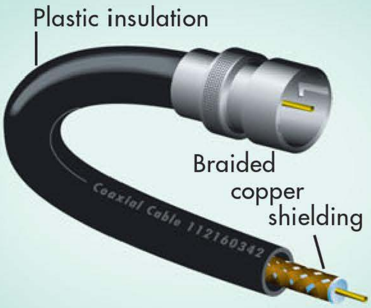



**Star topology** The most common shape for a LAN is the star. Several nodes are linked to the network by a cable, radio signals, optic fiber, and so on. They lead to a common point at the center of the star, where there's a hub, switch, or router (which are explained in the next illustration). Data from a node travels to the center of the star, where the device located there passes the data along to the node to which the data's addressed. Star configurations are often used to connect two LANs.

# How Network Nodes Connect

To become part of a network, a personal computer uses a **network interface card (NIC) or an RJ-45 connector that's part of the motherboard**. (For portable computers, the interface can be in the form of a PC Card or USB adapter.) Communications signals pass from the PC's RAM and through the connection to a LAN's **backbone**, the part of the network that carries the most traffic. The backbone and connections leading to and from it might use **coaxial** cable, **fiber-optic** cable, **twisted-pair** cable, **radio waves**, and phone and power wiring to link PCs. The combination of connector, circuitry, wiring, and other hardware determines the network's bandwidth.




## Coaxial Cable

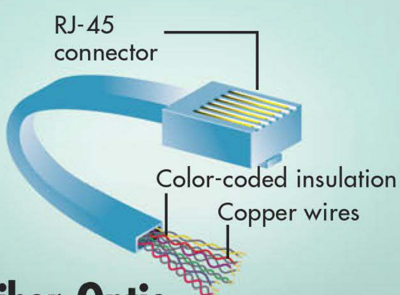
From the connector, data can be sent along **BNC coaxial cable**, like that used for cable television. (*BNC* stands for *Bayonet Neill-Concelman*, a fact you will not be quizzed on.) Coaxial consists of a single copper wire, which is sheathed by plastic and braided copper that shields the center wire from electrical disturbances. Each end of a segment of cable has a **bayonet connector**, which requires only a quarter of a turn to attach the cable.

## Twisted-Pair Wiring

A more common alternative to coaxial is twisted-pair wiring. A plastic outer jacket encloses four pairs of insulated wire that are twisted with a different number of turns per inch. The twists cancel out electrical **noise** from adjacent pairs of wires and from motors and other electrical devices in the same building.

Each end of the cable terminates in a plastic **RJ-45** connector, which resembles the common RJ-11 phone plug. (*RJ* stands for *registered jack*.)

Each node on the network has a separate twisted-pair cable that connects the node to a central **hub**, **router**, or **switch**, which is the center of a star configuration. All of these devices let the signals from any one computer travel to any other node on the network. Any of the connections can be broken without affecting other nodes.

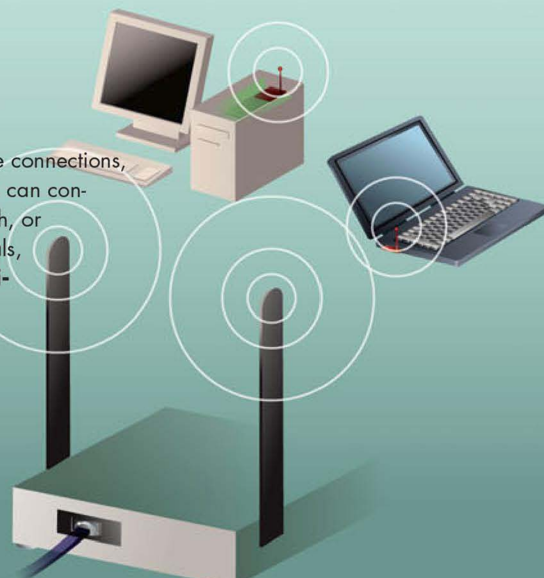


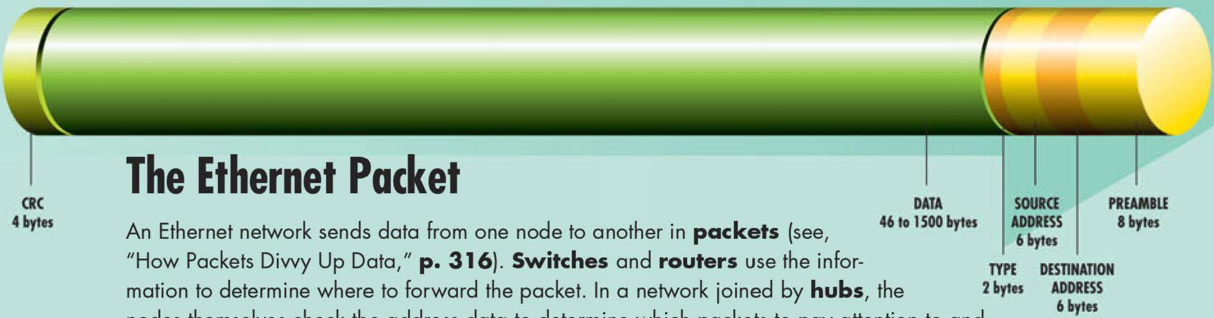

## Fiber-Optic

On networks connecting directly to the Internet or in LANs for which speed is crucial, **fiber-optic** cable carries 1 billion bits a second, enough to carry tens of thousands of telephone calls. Hair-thin fibers consist of two layers of pure silica glass covered with a reflective **cladding**, like a tunnel lined with mirrors. Varying pulses of light from a laser or LED carry the data along the twists and turns of the cable by bouncing off the cladding.


## Wireless

Instead of using cable connections, nodes on the network can connect to the hub, switch, or router via radio signals, such as those that **Wi-Fi** systems use (see Chapter 29). In fact, all of the connection methods described here can be used together on the same LAN.






## The Ethernet Packet

An Ethernet network sends data from one node to another in **packets** (see, "How Packets Divvy Up Data," **p. 316**). **Switches** and **routers** use the information to determine where to forward the packet. In a network joined by **hubs**, the nodes themselves check the address data to determine which packets to pay attention to and which to ignore.

- **Preamble**—Synchronizes the network nodes.
- **Destination Address**—A single PC or all PCs on a network.
- **Source Address**—The address of the computer from which the packet originated.
- **Type**—Defines the format used for the data.
- **Data**—The actual information.
- **CRC**—Cyclical Redundancy Check, which is used to spot transmission errors.

# Hubs, Routers, and Switches

In a star configuration, a network uses hubs, switches, and/or routers as traffic cops to move data to the right destination and to ward off intruders from the Internet. Each of these devices is a simple box with several plugs to accept RJ-45 or fiber-optic cables.

### Hubs

1. A hub receives incoming data packets from different nodes and temporarily places them in a memory buffer if the hub is busy with another packet.

2. Each packet the hub receives is sent to every other node regardless of the packet's addressing. Nodes ignore any packets that are not addressed to them.

### Switches

1. A switch functions similarly to a hub, but a switch knows which of its connections lead to specific nodes. The switch reads a packet's addressing information and transmits the packet out only on the line that leads to the node it's addressed to.

2. Some packets—for example, one announcing that another computer has come online—arrive addressed for **broadcast**. This means the sending node wants all other nodes to see the packet. The switch sends copies of the packet.

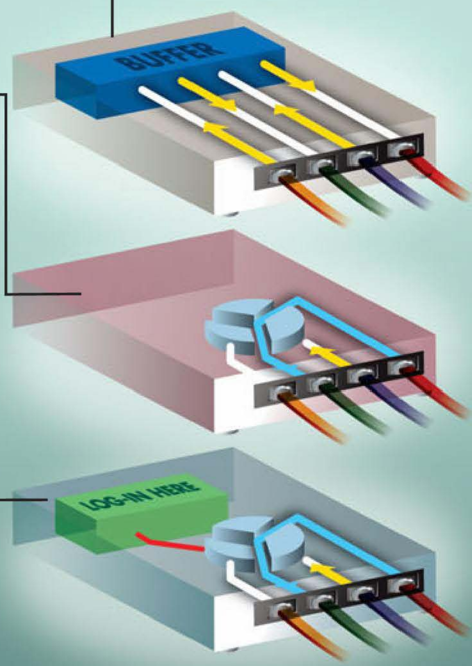
### Routers

1. A router is similar to a switch, except that a router does not accept or transmit broadcast packets. A router requires a specific delivery address for a node located on the LAN. (But most routers also have switch capability.)

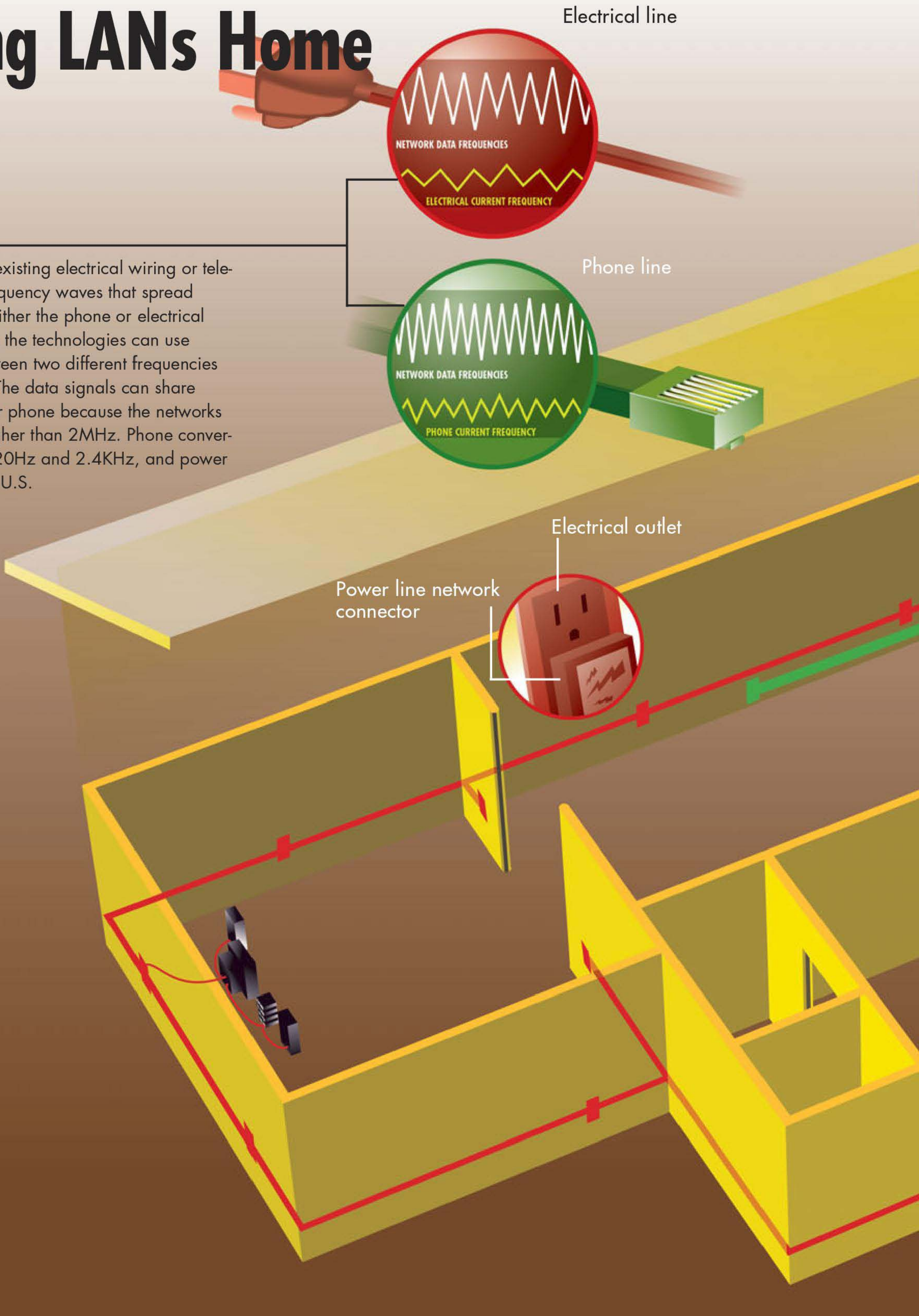
2. Routers provide connections to the Internet at the same time they protect the LAN from the Internet. The rules might, for example, require the router to block any LAN packet that has a destination address outside the LAN and somewhere in the Internet.

3. If the packet comes from the Internet and is headed toward a node on the LAN, the router can send the signal to a log-in routine or reject it entirely.

4. If the destination address is valid—say, for an email server on the LAN—the router lets the packet into the network. Before sending the data to its destination, some routers check the packet's CRC segment for errors that have occurred en route. If a packet has an error, the router discards it and sends a message to the origination address, requesting a fresh packet of the same data.

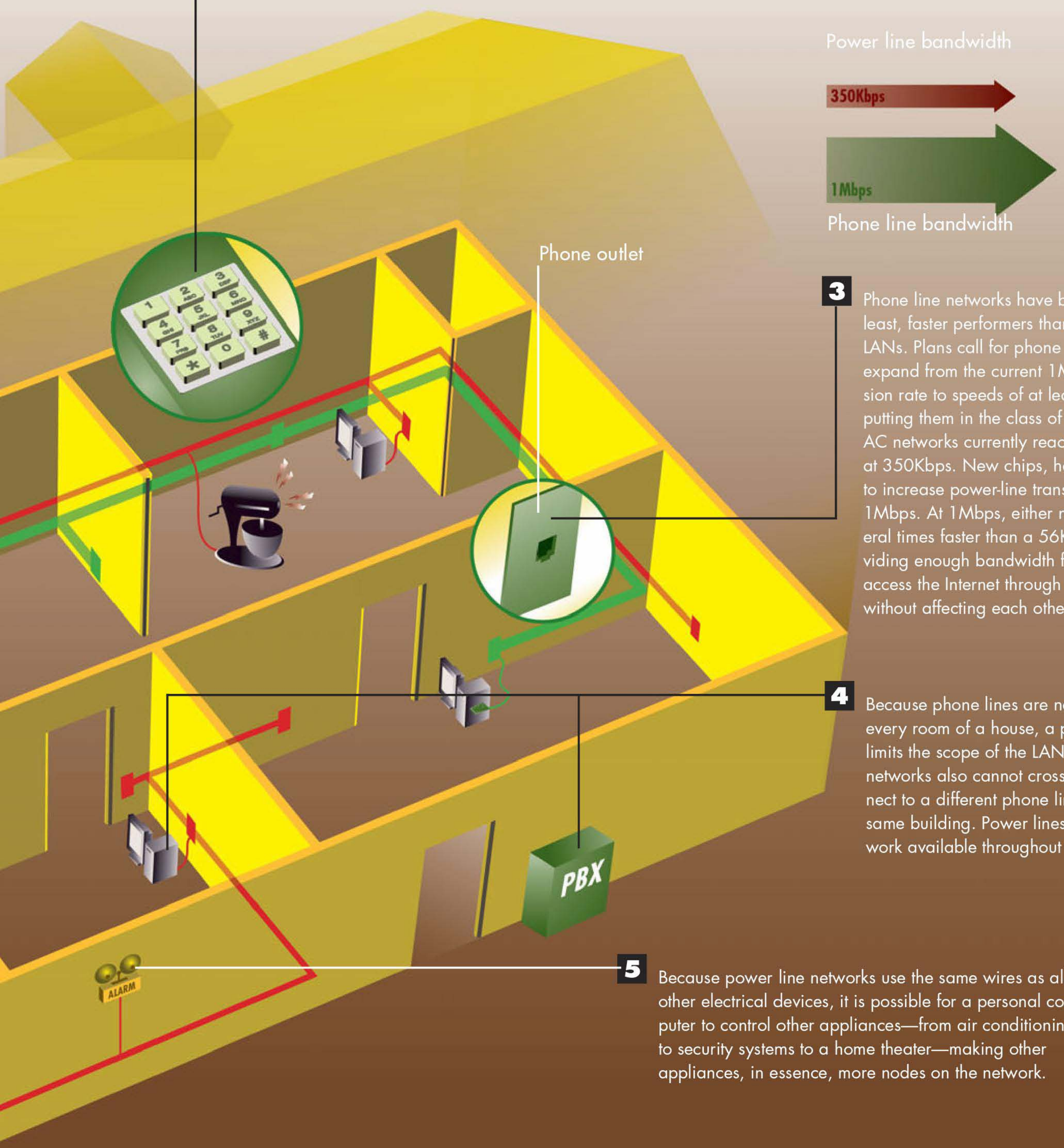


# How Phone and Power Lines Bring LANs Home




**1** Networks that work over existing electrical wiring or telephone wires use radio frequency waves that spread through all the wiring in either the phone or electrical systems. The variations on the technologies can use on/off pulses or shift between two different frequencies to represent bits of data. The data signals can share wires used for electrical or phone because the networks operate at frequencies higher than 2MHz. Phone conversations operate between 20Hz and 2.4KHz, and power lines cycle at 60Hz in the U.S.

**2** Because other uses of the lines dramatically affect their electrical characteristics, both networks must adapt instantly to changes in current or voltage. By avoiding the frequencies involved in electrical current and phone signals, electrical noise does not contaminate data on the lines. Phone conversations, fax transmissions, and use of appliances plugged into power lines continue normally without affecting or being affected by the network transmissions.



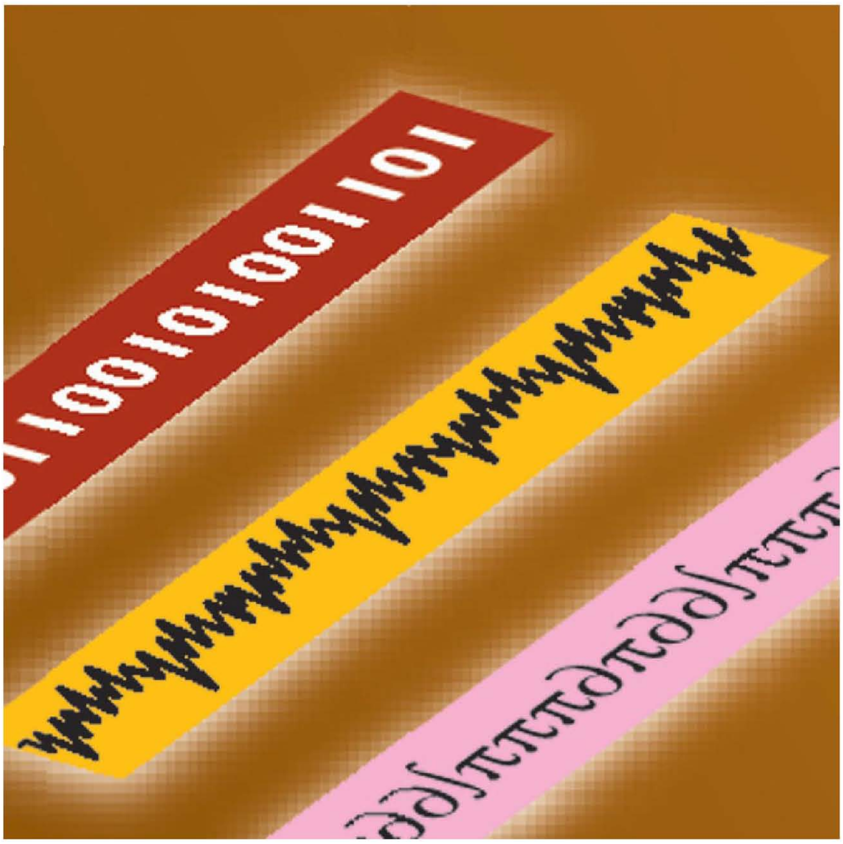

**3** Phone line networks have been, initially at least, faster performers than power-line LANs. Plans call for phone networks to expand from the current 1Mbps transmission rate to speeds of at least 300Mbps, putting them in the class of a slow Ethernet. AC networks currently reach a speed limit at 350Kbps. New chips, however, promise to increase power-line transmissions to 1Mbps. At 1Mbps, either network is several times faster than a 56K modem, providing enough bandwidth for several PCs to access the Internet through a single link without affecting each other's performance.

**4** Because phone lines are not always in every room of a house, a phone network limits the scope of the LAN. Phone line networks also cannot cross a PBX or connect to a different phone line in the same building. Power lines make a network available throughout a building.

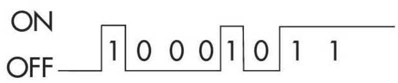
**5** Because power line networks use the same wires as all other electrical devices, it is possible for a personal computer to control other appliances—from air conditioning to security systems to a home theater—making other appliances, in essence, more nodes on the network.

CHAPTER 25

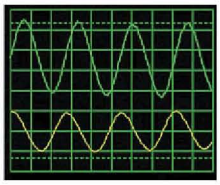
# How PCs Connect to the Internet



**YOUR** PC is a digital device. It accomplishes most of its tasks by turning a series of electronic switches on or off. A binary 0 represents a switch that is turned off; a binary 1 indicates that the switch is on. There is no in-between designation. A graph of digital code would look like this:




The traditional telephone system is an analog device, designed—at a time when digital electronics was unknown—to transmit the diverse sounds and tones of the human voice. Those sounds are conveyed electronically, in an analog signal, as a continuous electronic current that smoothly varies its frequency and strength. It can be depicted on an oscilloscope as a wavy line, such as this:



A **modem** is the bridge between digital and analog signals. It converts on and off digital data into an analog signal by varying, or **modulating**, the frequency of an electronic wave, a process similar to what FM radio stations use. On the receiving end of a phone connection, a modem does just the opposite: It **demodulates** the analog signals back into digital code. The two terms *modu*late and *dem*odulate give the modem its name.

There are several modems that aren't modems at all because they work with digital signals. Among them are **cable modems**, which connect a PC to the Internet using the same pathways as cable television, and **digital subscriber line (DSL)** modems, which send purely digital signals over telephone lines. The more correct term for them is **terminal adapters**, but no one calls them that. So we won't either.

Cable and DSL make surfing the Internet a totally different experience. They are **broadband**, which means they have the **bandwidth**—the capacity—to carry so many words, images, and video to your computer, so quickly, that the Net becomes virtually instantaneous 24/7. No waiting, no cussing. The appeal is so great that within a few years, the dial-up modem will be found only where cable and DSL are too distant to work. And even in the wilderness, there's the option of employing a broadband satellite connection that can send broadband anywhere.

We'll look at both DSL and cable connections in this chapter, but first we look at the traditional modem, which manages data transfers up to 28.8 kilobits a second, though most dial-up modems today use a scheme called **V.90**, which involves both analog and pure digital communications for transfer rates nearly twice as fast as conventional modems.

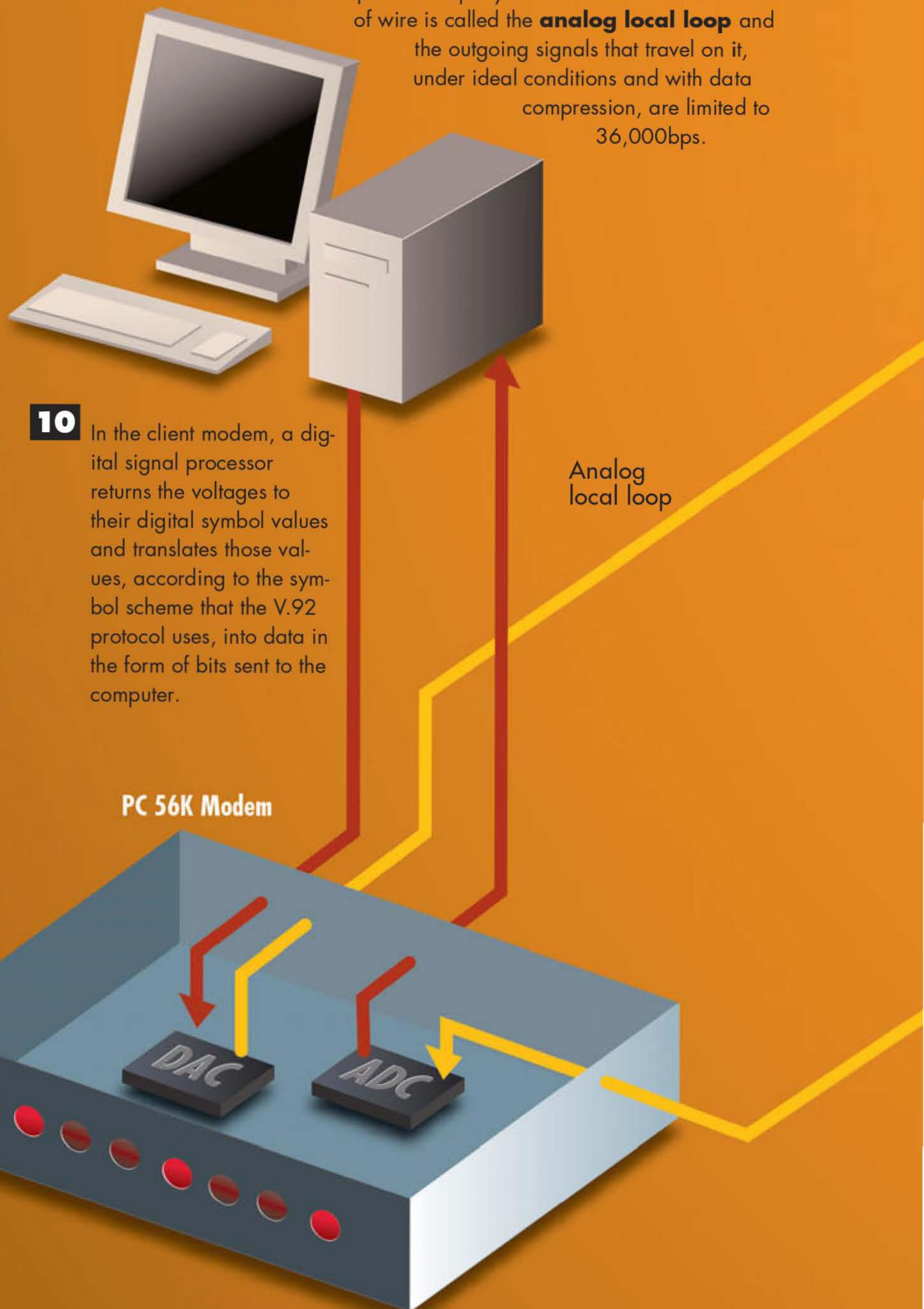
A lot of people continue to use these ancient dial-up connections. If you don't, it's still worthwhile knowing how they work. Knowing about the first modems gives you a better understanding and appreciation for the newest modems.

# How a Dial-up Modem Calls the Internet

**2** The PC sends a message or a request, which begins as digital data in the PC. The modem converts the bits to an ordinary telephone analog signal. The request travels from the PC's modem along two twisted copper wires that lead to a local telephone company's **central office**. This stretch of wire is called the **analog local loop** and the outgoing signals that travel on it, under ideal conditions and with data compression, are limited to 36,000bps.


**1** The current standard in dial-up modem technology is called 56K, or V.92. The technology allows a person at a home or office PC to send data at the normal analog modem rates of 33,600bps (bits per second), but get information back nearly twice as fast. When a PC uses a 56K modem to connect to a **host**, the modem first probes the connection to determine whether the host modem supports the V.92 standard of transmitting 56,600bbps and whether any of the signals coming from the host to the PC go through any conversions from digital to analog to digital again. In the case of multiple conversions, 56K transmissions are not possible, and the modems on either end of the connection default to normal transmissions. However, if a 56K connection is possible, the PC and host modems synchronize their timing to help ensure the accuracy of the downloaded data.

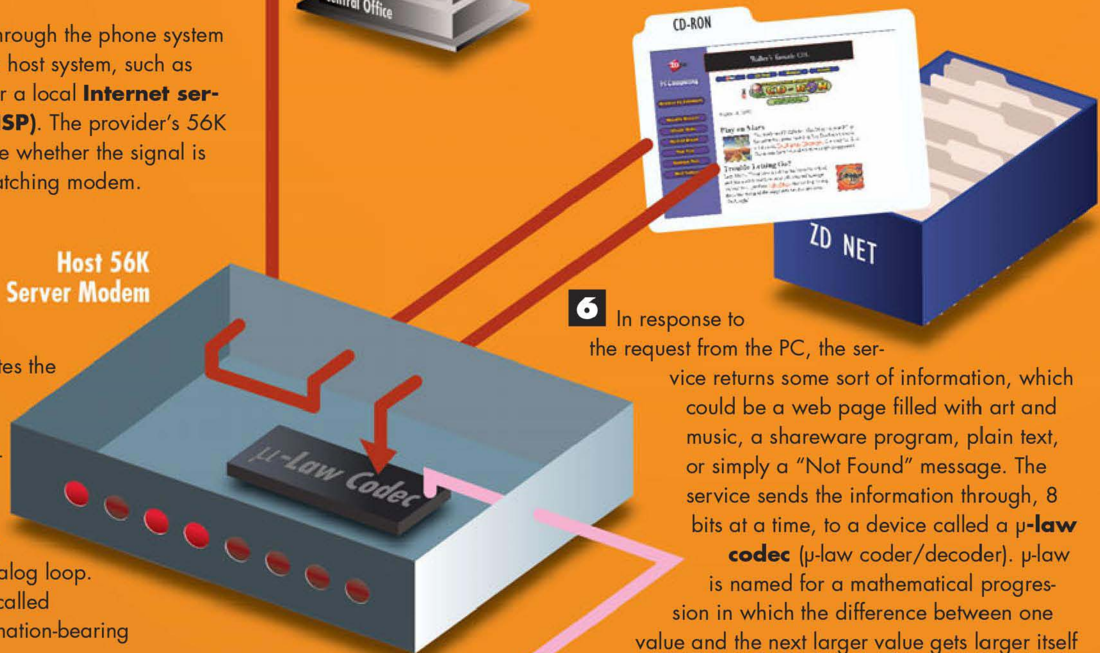
**10** In the client modem, a digital signal processor returns the voltages to their digital symbol values and translates those values, according to the symbol scheme that the V.92 protocol uses, into data in the form of bits sent to the computer.




**3** At the central office, the signal enters the **public switched telephone network (PSTN)**, what we generally think of as simply "the telephone system." There, the analog signal is transformed into a digital signal. Except for the local analog loops, virtually the entire U.S. phone system transmits data digitally, which theoretically allows data to move as fast as 64,000bps and makes V.92 technology possible.

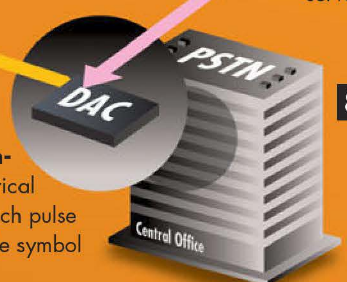
**4** At the telephone facility, an **analog-to-digital converter (ADC)** samples the incoming analog waveform 8,000 times a second, and each time the signal's approximate amplitude, or the strength of the current, is recorded as an 8-bit **pulse code modulation (PCM)**. Although there are an infinite number of values for the amplitude each time it's sampled, the ADC can detect only 256 discrete levels. That's because along the path the electronic signals travel, it encounters fleeting electromagnetic fields that can come from innumerable sources, including the phone wires themselves. These fields produce a current voltage that distorts the signal's value—makes it fuzzy. This **line noise** limits analog lines to 33.6Kbps.


**5** The data travels through the phone system until it arrives at a host system, such as America Online or a local **Internet service provider (ISP)**. The provider's 56K modems recognize whether the signal is coming from a matching modem.




**6** In response to the request from the PC, the service returns some sort of information, which could be a web page filled with art and music, a shareware program, plain text, or simply a "Not Found" message. The service sends the information through, 8 bits at a time, to a device called a **µ-law codec** (µ-law coder/decoder). µ-law is named for a mathematical progression in which the difference between one value and the next larger value gets larger itself as the values increase. (In Europe, an A-law codec uses a different coding scheme, which is why U.S. 56K modems only reach their full potential communicating with servers in the U.S.)

**7** The codec translates the data into values corresponding to the most recognizable of the 256 possible voltages that can be distinguished on an analog loop. These values are called **symbols**—information-bearing tokens.

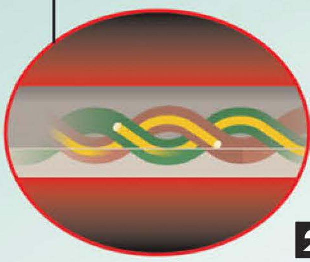


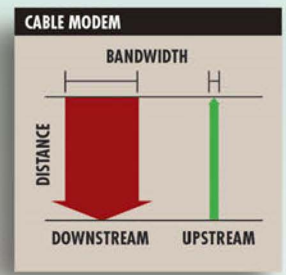

**8** If the modems could use all 256 possible symbols, they could send 64,000 bits of data each second. But the host and client modems use some of the symbols to keep tabs on each other. The symbols whose values approach zero are too closely spaced to each other to be distinguished accurately on a noisy phone line. The modems rely on the 128 most robust symbols, which only allow 56,600bps. (Current FCC regulations limit the power that can be used for phone signals, which limits real-world transmission rates to about 53Kbps.)

**9** When the signals reach the analog local loop, a **digital-to-analog converter (DAC)** generates 8,000 electrical pulses each second. The voltage of each pulse corresponds to one of the 128 possible symbol values.

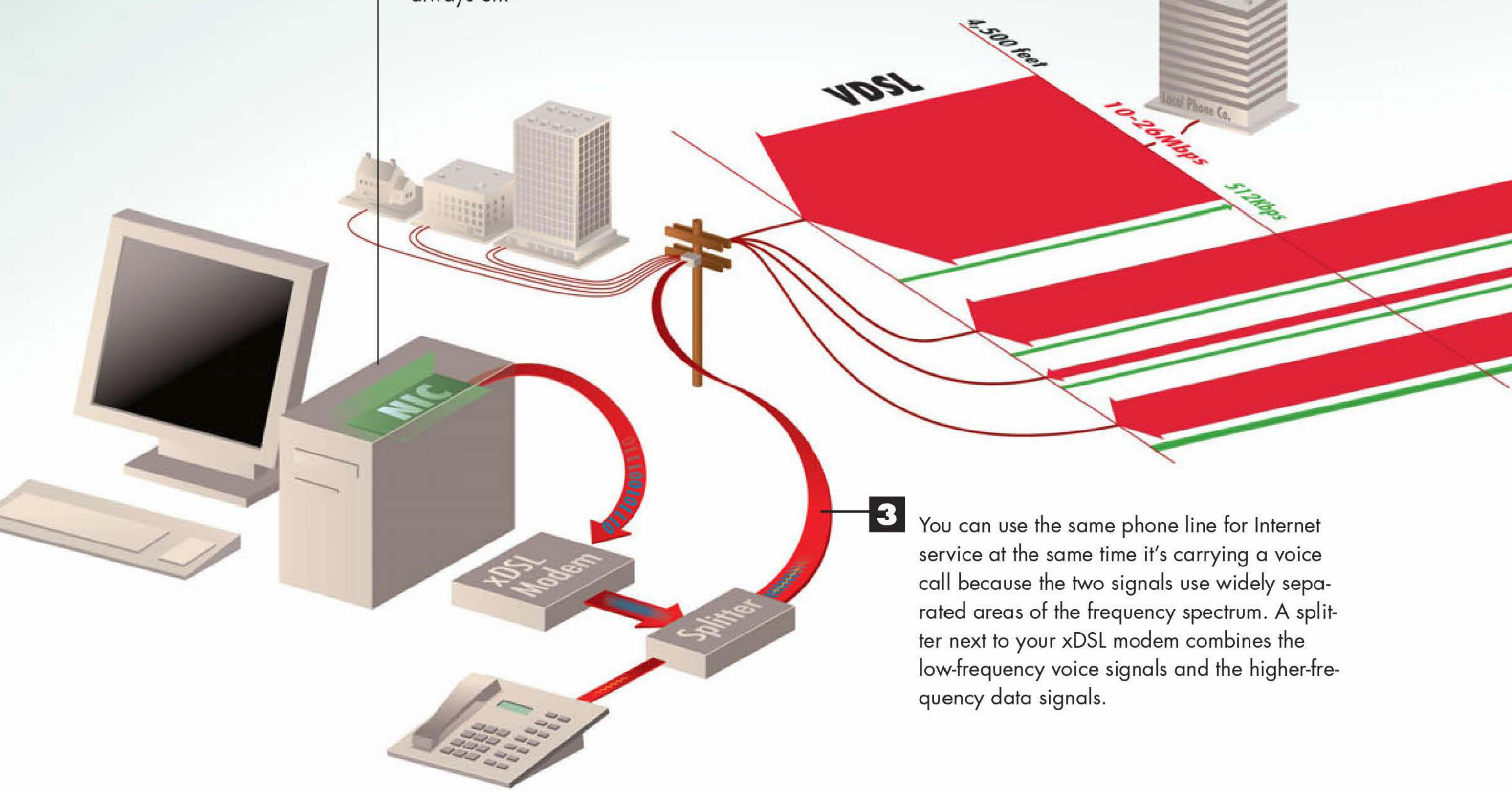
# How DSL Turbocharges a Phone Line

**1** There are several forms of **digital subscriber lines**, or **xDSL**, with the *x* depending on the particular variety of DSL. All xDSL connections use the same ordinary pair of twisted copper wires that already carry phone calls among homes and businesses. Unlike cable modem connections, which broadcast everyone's cable signals to everyone on a cable hub, xDSL is a point-to-point connection, unshared with others using the service.

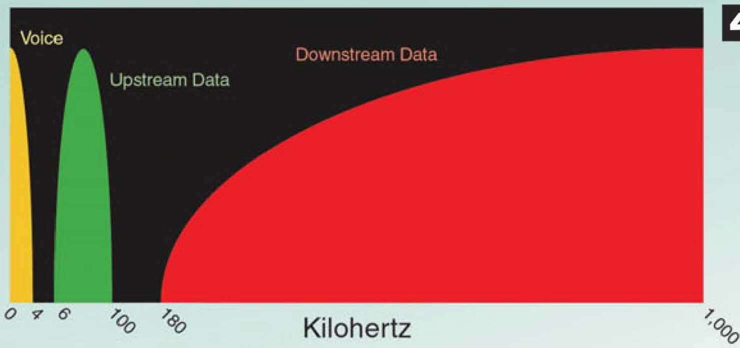





**2** Signals travel between a **coax** or **RJ-45** connector in your PC and an xDSL modem. You do not have to dial up an Internet service provider; your Net connection is always on.



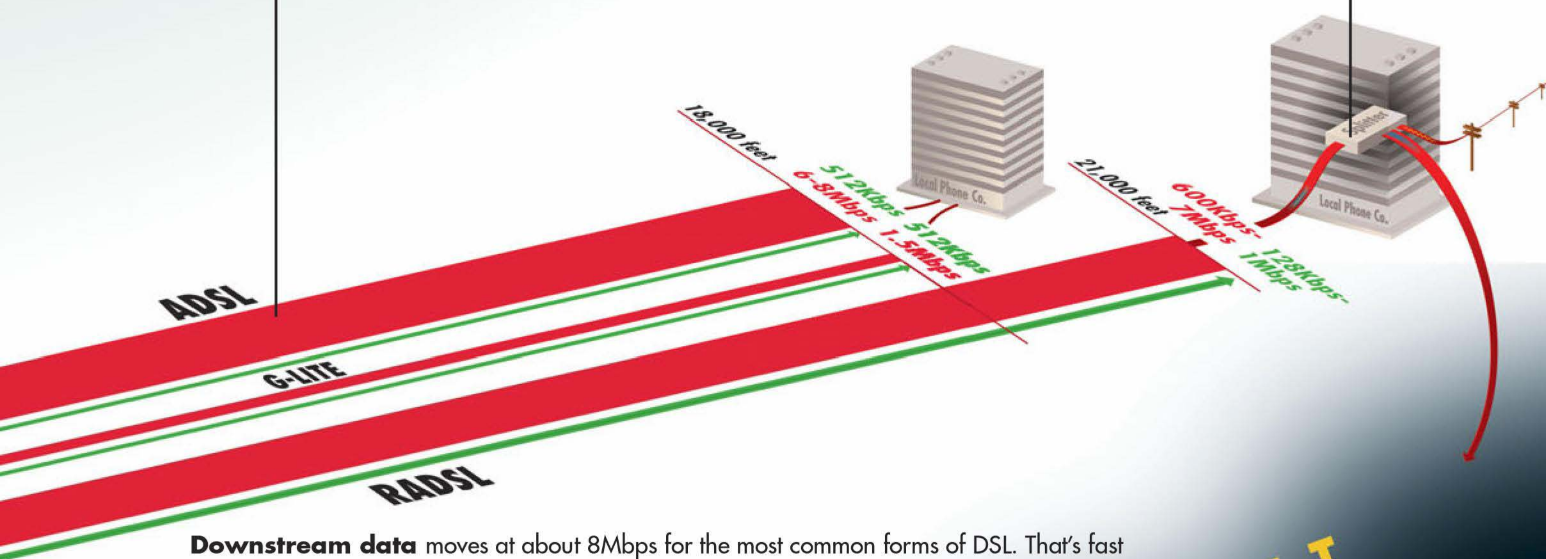

**3** You can use the same phone line for Internet service at the same time it's carrying a voice call because the two signals use widely separated areas of the frequency spectrum. A splitter next to your xDSL modem combines the low-frequency voice signals and the higher-frequency data signals.




**4** The most common form of xDSL is **ADSL**. The *A* stands for **asynchronous**, meaning that more bandwidth, or data-carrying capacity, is devoted to data traveling downstream—from the Internet to your PC—than to upstream data traveling from your PC to the Internet. The reason for the imbalance is that upstream traffic tends to be limited to a few words at a time—a URL, for example—and downstream traffic carrying graphics, multimedia, and shareware program downloads needs the extra capacity.

**5** Transmission rates depend on the quality of the phone line, the type of equipment it uses, the distance from the PC to a phone company switching office, and the type of xDSL being used.

**6** A splitter on the other end of the line breaks the voice and data signals apart again, sending voice calls into the **plain old telephone system (POTS)** and computer data through high-speed lines to the Internet.




**Downstream data** moves at about 8Mbps for the most common forms of DSL. That's fast enough to transmit Moby Dick in just under six minutes compared to the 14 hours a V.90 modem would take.

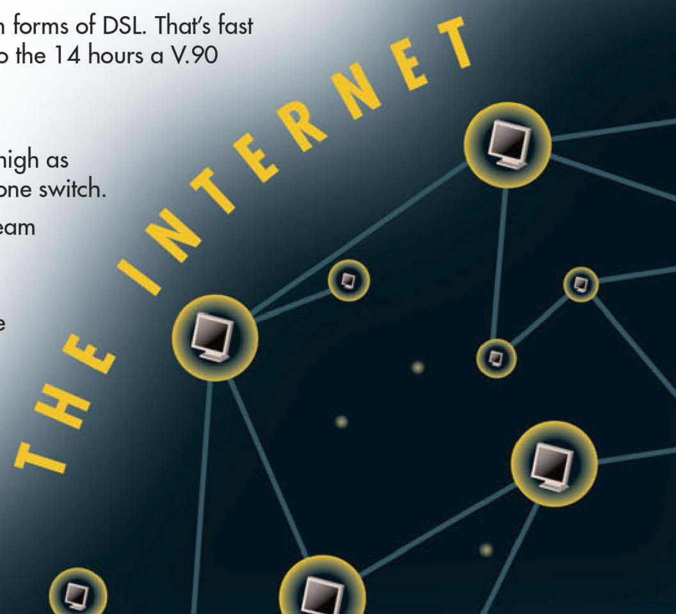
**Upstream data** moves at about 640Kbps.

**VDSL** (very high-speed DSL) reaches data transmission speeds as high as 10–26Mbps downstream, but only within about 4,500 feet of a phone switch.

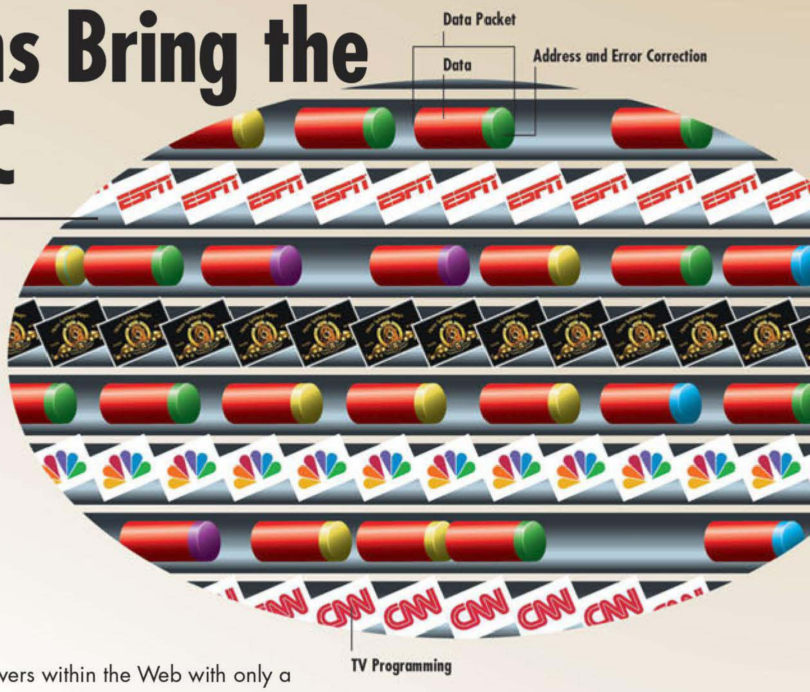
**ADSL** is limited to 18,000 feet from the phone office with downstream speeds of 6–8Mbps, upstream 512Kbps.

**G-Lite**, or **Universal DSL**, at the same 18,000 feet, allows only 1.5Mbps downstream and 512Kbps upstream. G-Lite eliminates the need for a splitter and is the industry's proposal for a standard modem that can work with any phone lines.

**RADSL** (rate adaptive DSL) reaches as far as 21,000 feet but is limited to 600Kbps–7Mbps downstream rates and 128Kbps–1Mbps upstream.

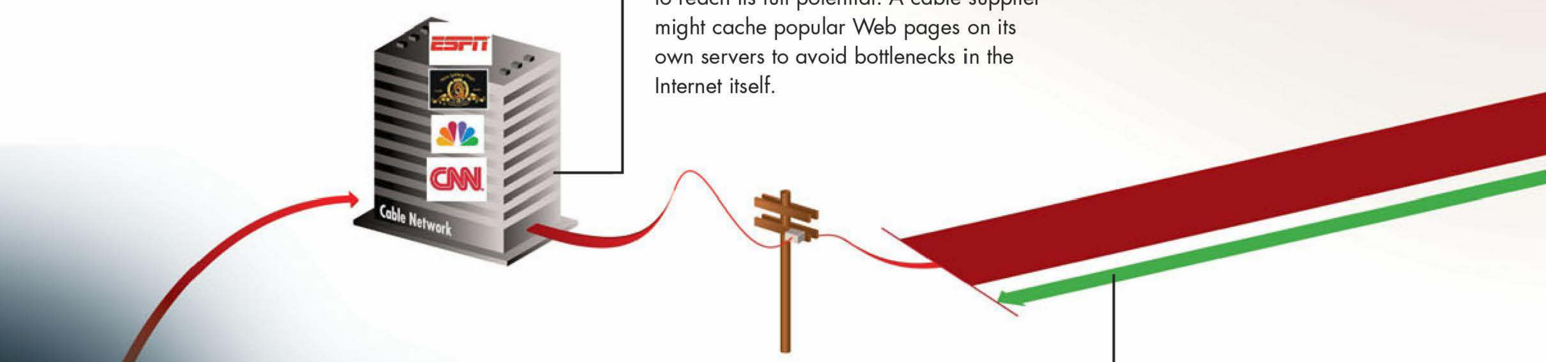


# How Cable Modems Bring the Internet to Your PC




**1** Computer data is sent along frequencies that lie between the 110 6MHz frequency bands carrying TV programming over **hybrid fiber coaxial (HFC)** cable. Data is sent as standard **Internet protocol (IP)** packets with error correction and a header identifying which subscribers' PCs they are destined to reach. The wide bandwidth of fiber optics, depending on the configuration, lets you receive downloads at up to 40Mbps, fast enough to download *Moby Dick* in about one minute, compared to six minutes for a DSL connection and 14 hours with a V.90 modem.

**7** Slow servers within the Web with only a 1.5Mbps T1 connection can act as bottlenecks, not allowing your cable connection to reach its full potential. A cable supplier might cache popular Web pages on its own servers to avoid bottlenecks in the Internet itself.

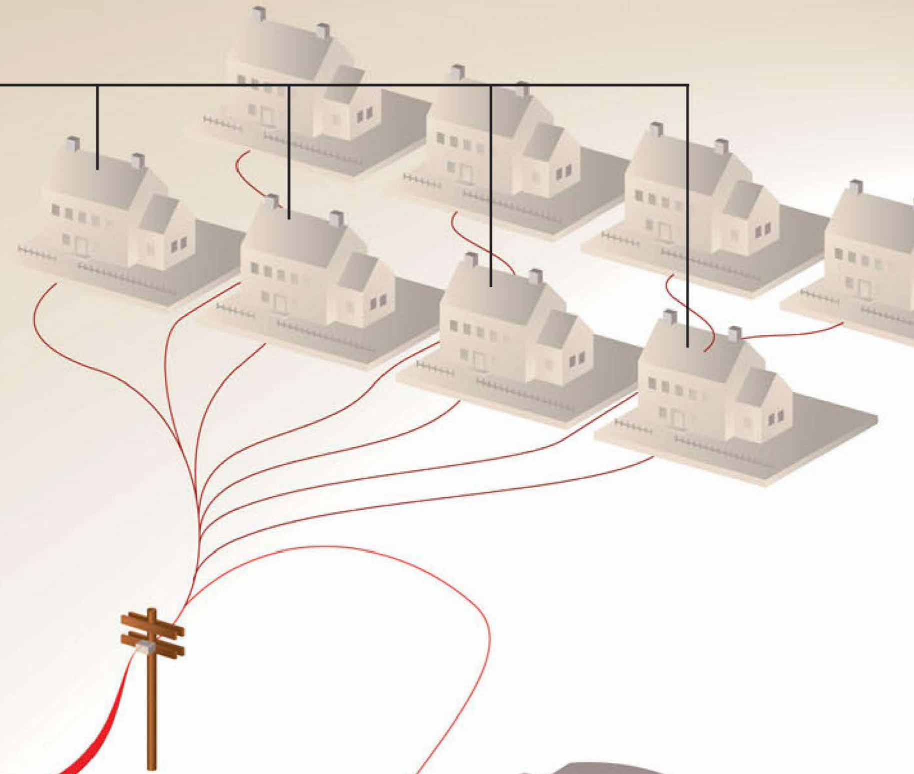


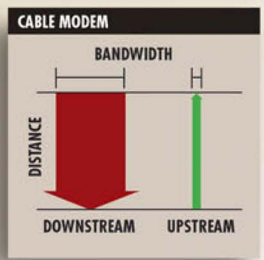


**5** Data going **upstream**—from your PC to the Internet—gets only seven bands for transmission, limiting upstream data to 30Mbps or lower.

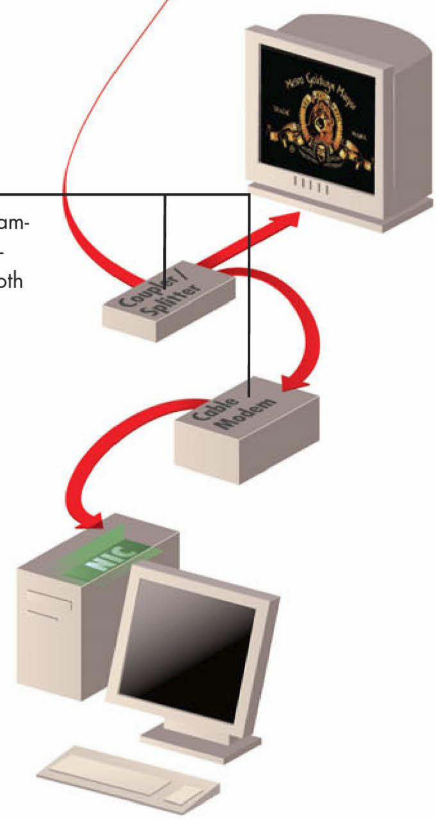
**6** High-speed fiber optic cable links each distribution hub to the cable supplier's own network, which handles email and routes PC data to the Internet through a 45Mbps T3 connection. The cable network is flexible so that if one part of it crashes, your data is routed through working sections of the network.

**2** Cable is a broadcast medium. Everyone connected to the same distribution hub receives everyone else's downloads, too. This arrangement raises security issues and could cause a slowdown if everyone connected to the same hub downloads at the same time.



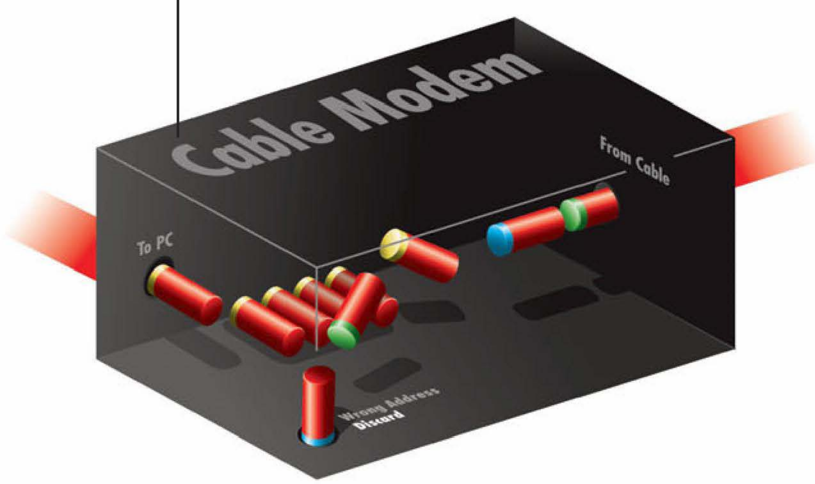






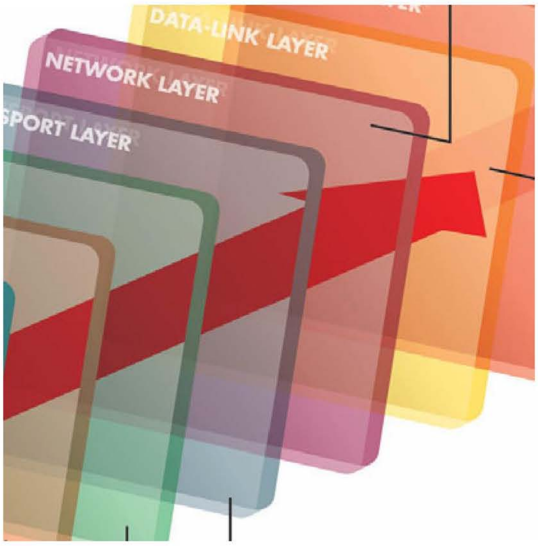

**3** A coupler/splitter routes TV programming to the television set and computer data to the cable modem. Both can be used simultaneously.

**4** Each cable modem ignores all packets except those addressed to it. The modem reassembles the packets bound for it into a coherent stream of data it sends to a network interface card in the PC.



CHAPTER 26

# How the Internet Moves Data



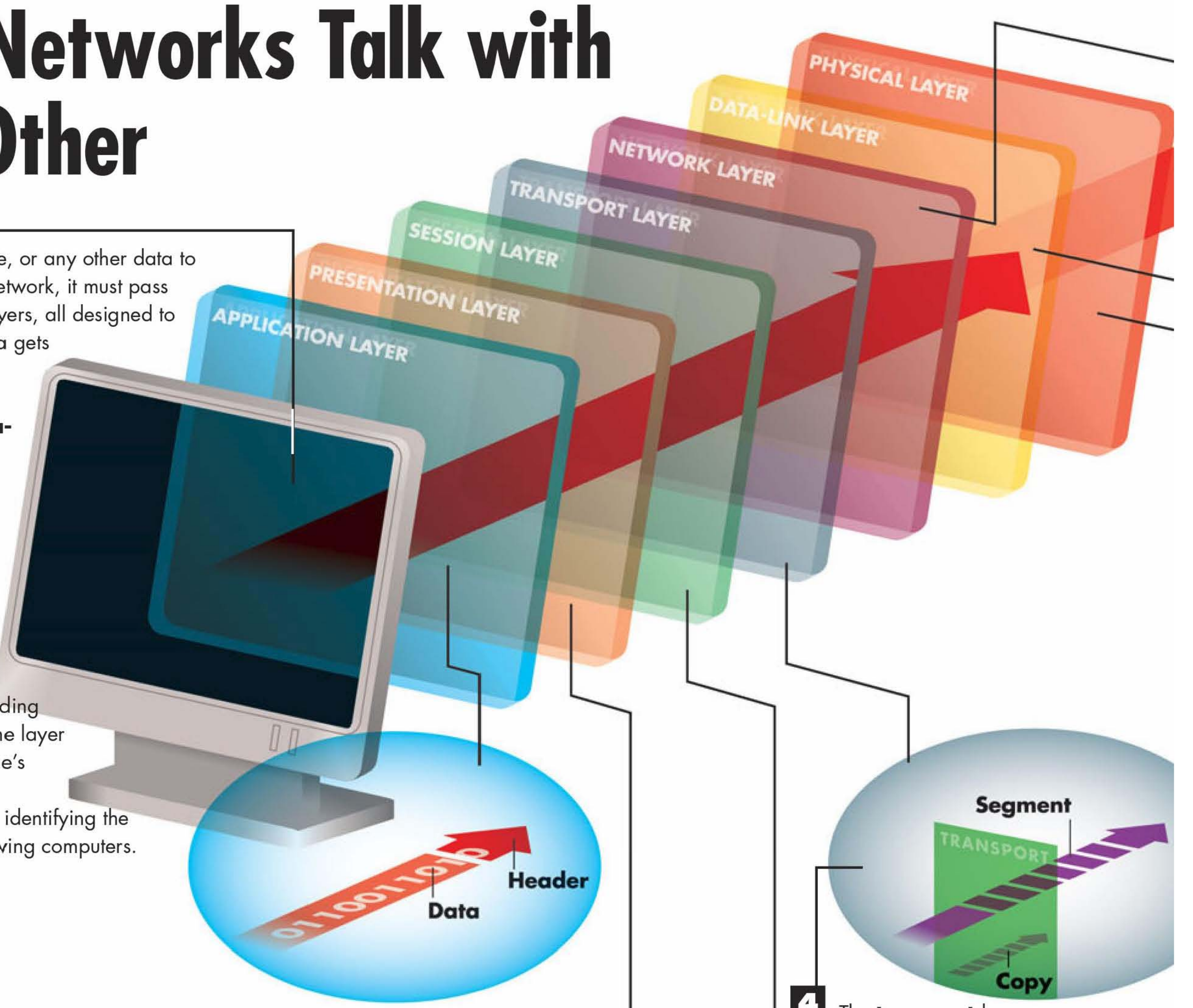
**WHAT'S** amazing about the Internet is not that it ties together the entire world into one giant conglomeration of civilization's accumulated data, but that your mom got the email you sent her yesterday.

Even the simplest communication over the Net is broken up into scores of discrete packages of data. Those packages are sent over thousands of miles of telephone lines, satellite signals, and microwaves—and all the packages in a single communication might not be sent over the same path. Along the way, they pass through systems running operating systems as diverse as UNIX, NetWare, Windows, and AppleTalk, through PCs, mainframes, and minicomputers—hardware and software that were never designed, really, to work peacefully with one another. Messages are coded and decoded, compressed and decompressed, translated and retranslated, bungled and corrected, lost and repeated, shredded and stitched back together again, all so many times that it's a miracle anything even resembling coherency gets from one place to the other, much less that it gets there filled with gorgeous fonts, musical fanfares, eye-popping photos, and dazzling animation.

Want to make it even more amazing? Consider this: No one's in charge! No one owns the Internet. Oh, people and companies own bits and pieces of it, and there are some organizations that by common consent rule over things such as the domain names that appear in addresses, such as www.quepublishing.com, www.whitehouse.gov, www.ucla.edu, and www.howcomputerswork.net. But their rule is entirely at the whim of the governed. There's no technological reason why someone couldn't start a specialized Net service—and in fact that's exactly what America Online, CompuServe, Microsoft Network, and Prodigy are, except that they're more than just themselves, because they are also portals to the whole of the Internet. There is, however, one good reason why someone would not start another, completely separate Internet—it'd be stupid. The thing that makes the Internet the Internet is its universal receptiveness. It's open to anyone with access to a computer and a connection. It includes more information, and misinformation, than you will learn in your lifetime.

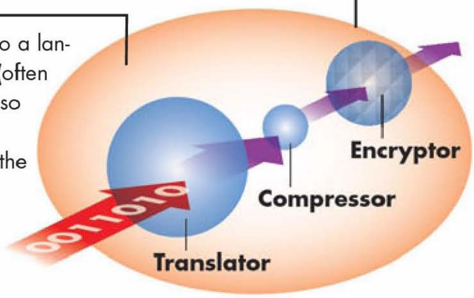
The Internet is built on cooperation. The illustrations in this chapter show how complex Internet communications are and how the technical challenges they present are overcome. The illustrations don't show an important element: the people behind these languages, routers, protocols, and contraptions that all cooperate to make sure you can send a dancing, singing email to Mom on her birthday.

# How Networks Talk with Each Other

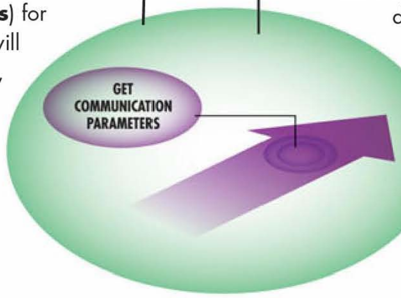



**1** For a message, file, or any other data to travel through a network, it must pass through several layers, all designed to make sure the data gets through intact and accurate. The first layer, the **application** layer, is the only part of the process a user sees, and even then the user doesn't see most of the work the application does to prepare a message for sending over a network. The layer converts a message's data into bits and attaches a header identifying the sending and receiving computers.

**2** The **presentation** layer translates the message into a language that the receiving computer can understand (often ASCII, a way of encoding text as bits). This layer also compresses and perhaps encrypts the data. It adds another header specifying the language as well as the compression and encryption schemes.

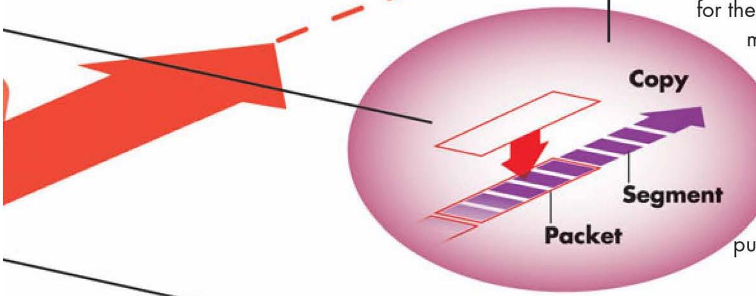


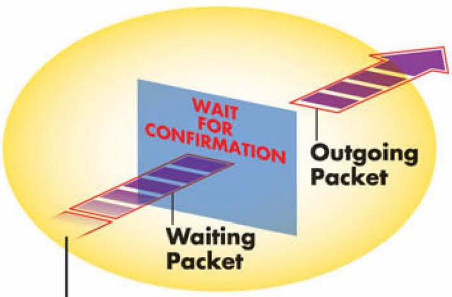

**3** The **session** layer opens communications. It sets boundaries (called **brackets**) for the beginning and end of the message and establishes whether the message will be sent **half duplex**, with each computer taking turns sending and receiving, or **full duplex**, with both computers sending and receiving at the same time. The details of these decisions are placed into a session header.




**4** The **transport** layer protects the data being sent. It subdivides the data into segments and creates checksum tests— mathematical sums based on the contents of data—that can be used later to determine whether the data was scrambled. It also makes backup copies of the data. The transport header i dentifies each segment's checksum and its position in the message.

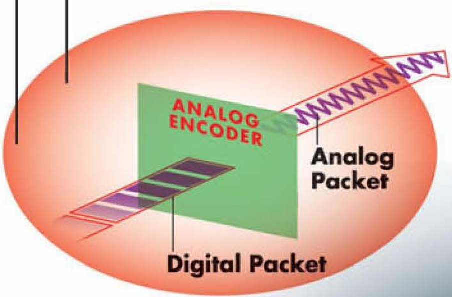
**5** The **network** layer selects a route for the message. It forms segments into packets, counts them, and adds a header containing the sequence of packets and the address of the receiving computer.



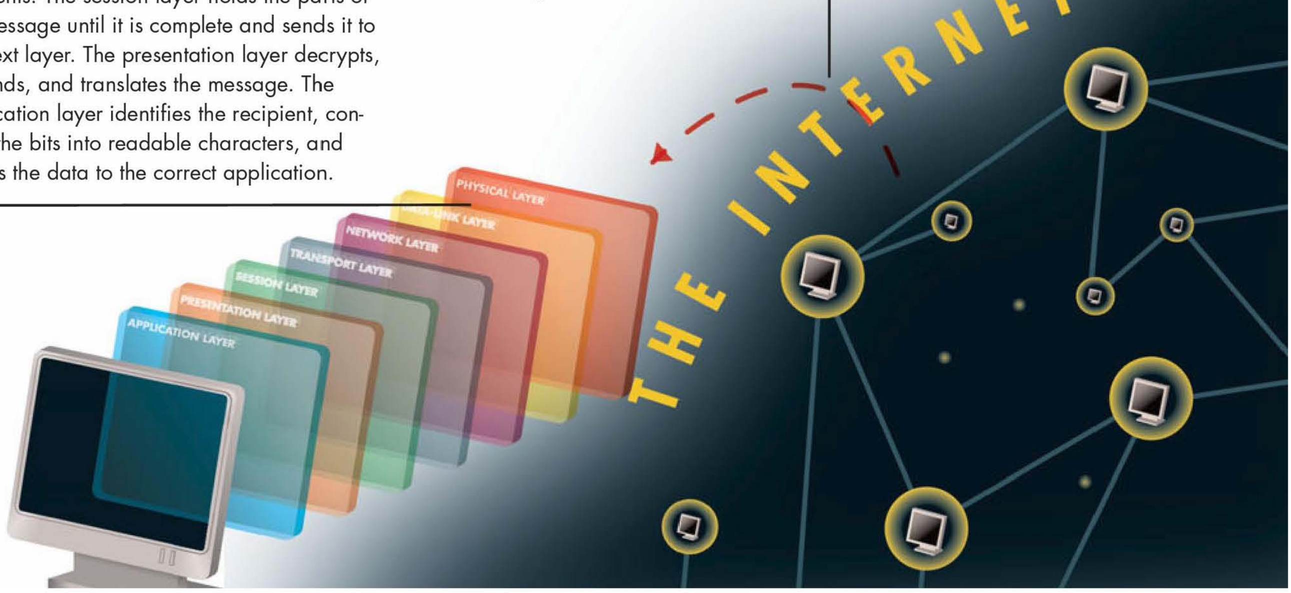

**6** The **data-link** layer supervises the transmission. It confirms the checksum, and then addresses and duplicates the packets. This layer keeps a copy of each packet until it receives confirmation from the next point along the route that the packet has arrived undamaged.

**7** The **physical** layer encodes the packets into the medium that will carry them—such as an analog signal, if the message is going across a telephone line—and sends the packets along that medium.




**9** At the receiving node, the layered process that sent the message on its way is reversed. The physical layer reconverts the message into bits. The data-link layer recalculates the checksum, confirms arrival, and logs in the packets. The network layer recounts incoming packets for security. The transport layer recalculates the checksum and reassembles the message segments. The session layer holds the parts of the message until it is complete and sends it to the next layer. The presentation layer decrypts, expands, and translates the message. The application layer identifies the recipient, converts the bits into readable characters, and directs the data to the correct application.

**8** An intermediate node calculates and verifies the checksum for each packet. A router might also reroute the message to avoid congestion on the network.



# How Information Travels the Internet

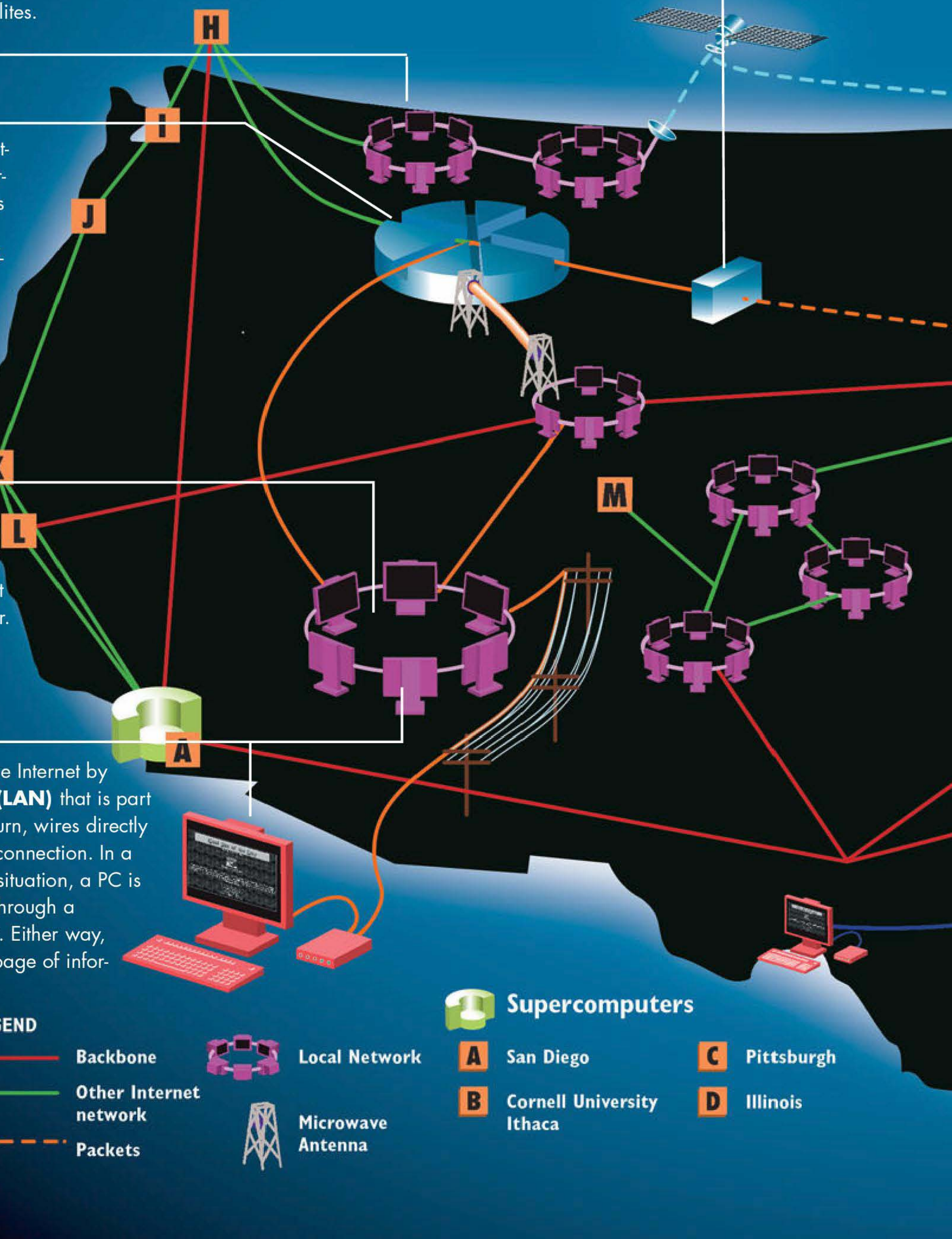
**1** In the office, a PC typically jacks in to the Internet by being part of a **local area network (LAN)** that is part of the greater Internet. The network, in turn, wires directly to the Internet through a port called a T-connection. In a **small office/home office (SOHO)** situation, a PC is more likely to use a modem to connect through a phone or cable connection to a network. Either way, through your browser you ask to see a page of information, and maybe multimedia, located on another computer somewhere else on the Internet.

**2** Your local host network makes a connection on another line to another network. If the second network is some distance away, your host LAN might have to go through a router.

**3** A **router** is a device that connects networks. It inspects your request to determine what other part of the Internet it's addressed to. Then, based on available connections and the traffic on different parts of the Net, the router determines the best path to set the request back on its track to the proper destination.

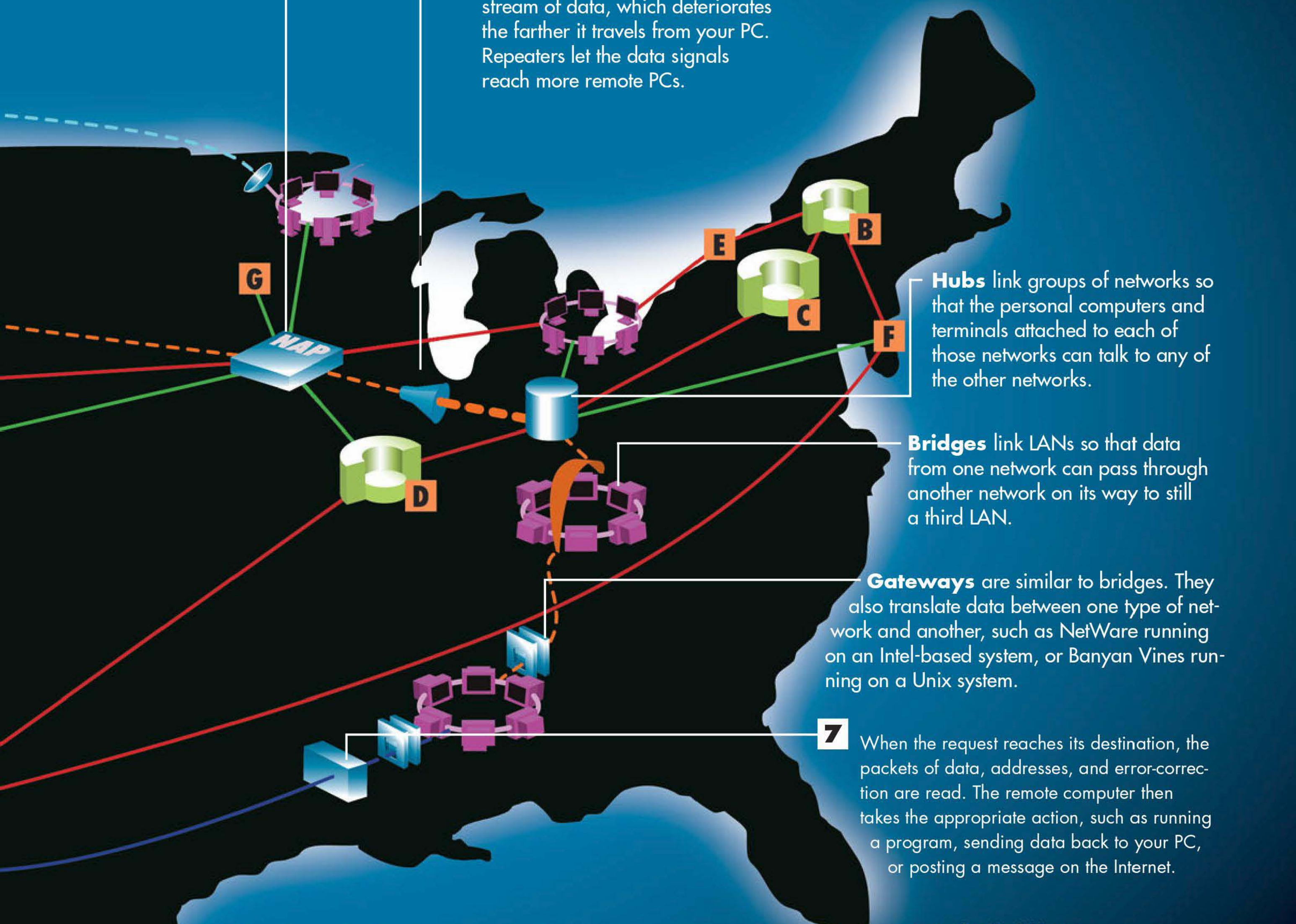
**4** Several networks in the same region might be grouped into a mid-level network. If your request is destined for another system within the same mid-level network, the router sends it directly to its destination. This is sometimes done through high-speed phone lines, fiber-optic connections, and microwave links. A variation, called a **wide-area network (WAN)**, covers a larger geographical area and can incorporate connections through orbiting satellites.

**5** As the request passes from network to network, a set of **protocols**, or rules, creates packets. Packets contain the data itself as well as the addresses, error checking, and other information needed to make sure the request arrives intact at the destination.



**6** If the destination for your request isn't on the same mid-level network or WAN as your host network, the router sends the request to a **network access point (NAP)**. The pathway can take any of several routes along the Internet's **backbone**, a collection of networks that link extremely powerful supercomputers associated with the National Science Foundation. The Internet, however, isn't limited to the United States. You can connect to computers on the Net in virtually any part of the world. Along the way, your request might pass through repeaters, hubs, bridges, and gateways.

**Repeaters** amplify or refresh the stream of data, which deteriorates the farther it travels from your PC. Repeaters let the data signals reach more remote PCs.




**Hubs** link groups of networks so that the personal computers and terminals attached to each of those networks can talk to any of the other networks.

**Bridges** link LANs so that data from one network can pass through another network on its way to still a third LAN.

**Gateways** are similar to bridges. They also translate data between one type of network and another, such as NetWare running on an Intel-based system, or Banyan Vines running on a Unix system.

**7** When the request reaches its destination, the packets of data, addresses, and error-correction are read. The remote computer then takes the appropriate action, such as running a program, sending data back to your PC, or posting a message on the Internet.

**Other important stops on the Internet**

- **E** SUNY at Buffalo
- **F** OCEANIC: University of Delaware
- **G** Minnesota Computer Center
- **H** University of British Columbia
- **I** University of Washington Information Navigator
- **J** PORTALS: Portland Area Library Services
- **K** Stanford Linear Accelerator
- **L** NASA Ames Research Center
- **M** Weather Underground, University of Colorado

CHAPTER 27

# How We Reach Each Other Through the Net

**ELECTRONIC** mail is everywhere. Many people in business, government, and education now use email more than the telephone to communicate with their colleagues. And in their private lives, many use email as an inexpensive but quick method of keeping in touch with friends and family scattered around the world.

Although electronic mail has been around since the formative years of the Internet, email first achieved mass popularity on local area networks. LAN-based email systems allow people in an office or campus to resolve problems without holding meetings, and to communicate with others without the dreariness of formal hard-copy memos. Today, the proprietary email systems of LANs are being replaced by email **client software** and **server software** used on the public Internet, creating popular, broad-based standards that make email easier for everyone.

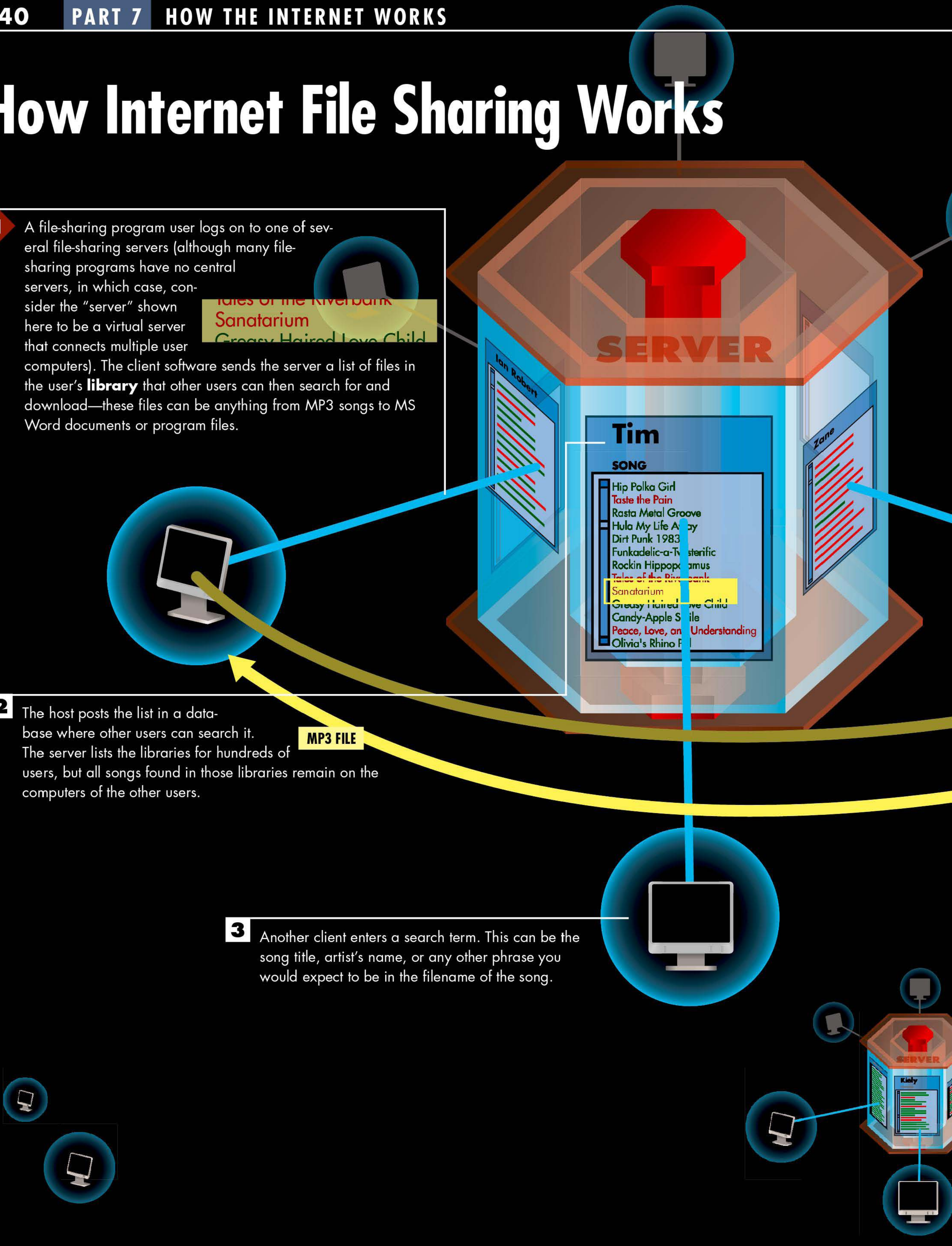
Internet email uses two main standards: the **Simple Mail Transfer Protocol**, or **SMTP**, to send messages, and the **Post Office Protocol**, or **POP**, to receive messages. Because these standards are universal, the software sending and receiving email, as well as the servers that handle the messages, can run on a variety of normally incompatible computers and operating systems.

The country's largest Internet service provider, America Online, doesn't use SMTP or POP, but instead uses its own proprietary protocols to send and receive email. The reason users of AOL can communicate with people outside AOL is that America Online uses gateway software that translates between different email protocols.

The body of an email message is where you type your note. But email has moved beyond mere words to encompass the capability to enclose complex document files, graphics, sound, and video. You can add these types of data to the body of a message by enclosing or attaching a file. When you enclose a file, your email software encodes it, turning all the multimedia data into ASCII text. At the receiving end, the user's software turns this ASCII gibberish back into meaningful data—if the software at both the user and the sender ends uses the same encoding scheme. The most popular of these is **MIME**, the **multipurpose Internet mail extensions**. MIME doesn't care what type of data is being enclosed with a message.

Another set of email standards can be seen as a great enabler of communication or an annoying source of inundating junk email—spam. Electronic mailing lists automatically send messages to a large number of users, either on a one-time basis or on a regular schedule. A **mail reflector** is server software that distributes mail to members of a mailing list. With a **list server**, individuals send email messages to subscribe and unsubscribe to the mailing list just as one would subscribe to a newsletter or magazine.

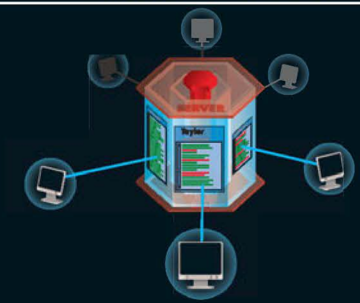
# How Internet File Sharing Works

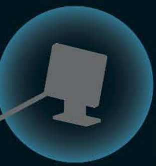



**1** A file-sharing program user logs on to one of several file-sharing servers (although many file-sharing programs have no central servers, in which case, consider the "server" shown here to be a virtual server that connects multiple user computers). The client software sends the server a list of files in the user's **library** that other users can then search for and download—these files can be anything from MP3 songs to MS Word documents or program files.

**2** The host posts the list in a database where other users can search it. The server lists the libraries for hundreds of users, but all songs found in those libraries remain on the computers of the other users.

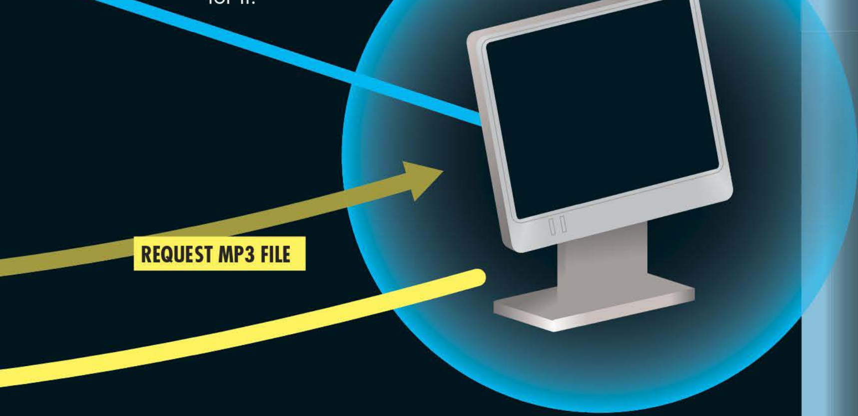
**3** Another client enters a search term. This can be the song title, artist's name, or any other phrase you would expect to be in the filename of the song.



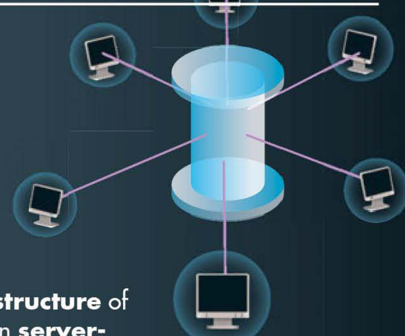


**4** The client looks at all the library records on the server and displays any titles that match at least some of the search criteria. The results include the names of the files, the type of Internet connection, **Internet protocol (IP)** addresses of other clients making those files available, and other assorted trivia.

**5** The user selects one or more of the files for transfer. His client software sends a message to the other client using its IP address. The message asks permission to download the song, and the remote client obliges by becoming a server and sending the file to the computer that asked for it.

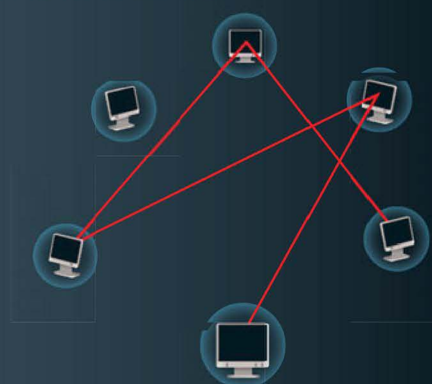



**6** At the same time someone is downloading a file, other users are finding songs on that person's hard drive and are downloading them. Several uploads/downloads can run simultaneously by taking turns sharing the Internet connections.



**The traditional structure** of the Internet has been **server-concentric**. Data and programs are stored at relatively few centrally located servers, or **hosts**. All data requests from PCs (**clients**) go to one of the servers. The host also handles all replies back to the clients.

**Distributed structure** is distributed. Except for consulting servers to get digital driving directions, each computer using the host software's protocol communicates directly with other clients. A PC can be a host and a client at the same time.



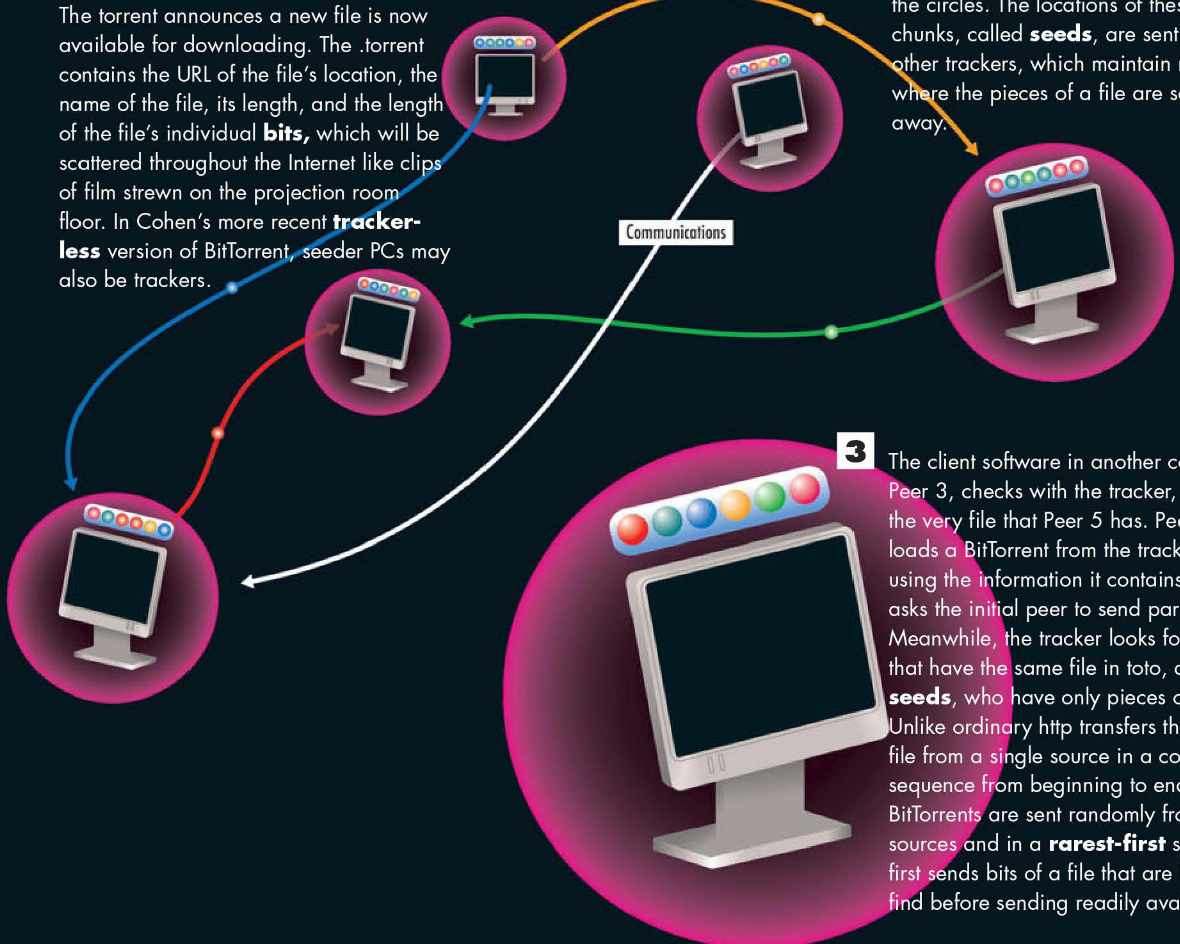
**True peer-to-peer structure:** No central host servers provide files for client PCs. Instead, the software is both host and client at the same time. Searches are more random, slower, and might turn up fewer hits. However, this structure makes it difficult, although not impossible, to police.

# How BitTorrents Spread the Wealth


It often seems as if the purpose of computers in the Grand Design of Things must be to provide free movies, songs, and software to the world's college students. The file sharing scheme that started it all, Napster, had a fatal flaw: The media companies could easily track down the computers on which everything from *Aida* to *Deep Throat* was laid out like a smorgasbord before ravenous downloaders. This forced file traders into new territory, **peer-to-peer** computing, that distributes data and culpability among millions of computer users. The most popular **P2P** dodge, the BitTorrent protocol, is estimated to account for a third of all internet traffic. And it all starts with a seed.

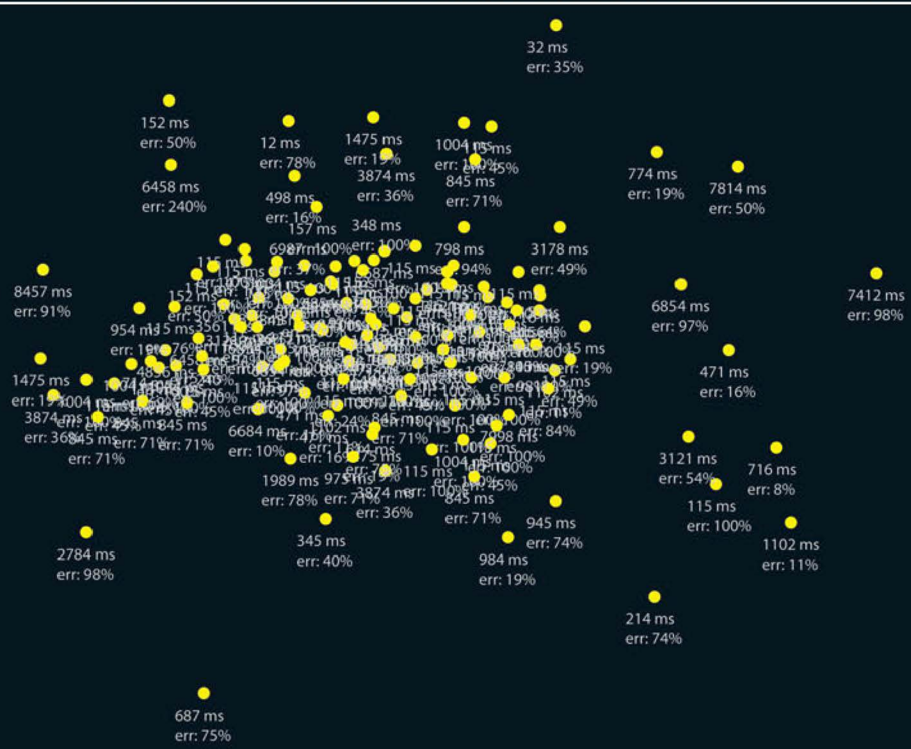
**1** A file, whether a video, song, book, or software, is placed on the Internet by the **initial seeder** using BitTorrent **client software**, such as Azureus, BitComet, or the original BitTorrent, invented in 2002 by programmer Brad Cohen. Through a system of open **ports** shared by computers, the seeder's client sends a small text file called a **.torrent** to a PC acting as a **tracker.** The torrent announces a new file is now available for downloading. The .torrent contains the URL of the file's location, the name of the file, its length, and the length of the file's individual **bits,** which will be scattered throughout the Internet like clips of film strewn on the projection room floor. In Cohen's more recent **trackerless** version of BitTorrent, seeder PCs may also be trackers.

**2** Here, a file is represented by the row of colored circles. Unlike Napster-era file - swapping, an entire file is not downloaded from a single source. Instead, a downloader receives the file in chunks—the circles. The locations of these chunks, called **seeds**, are sent to the other trackers, which maintain records of where the pieces of a file are squirreled away.

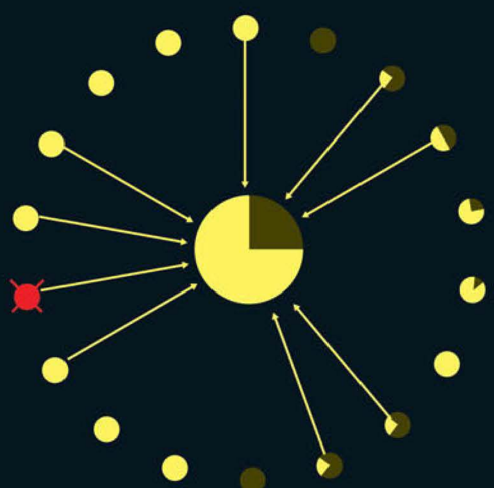



**3** The client software in another computer, Peer 3, checks with the tracker, looking for the very file that Peer 5 has. Peer 3 downloads a BitTorrent from the tracker, and using the information it contains, Peer 3 asks the initial peer to send part of the file. Meanwhile, the tracker looks for other peers that have the same file in toto, and for **seeds**, who have only pieces of the file. Unlike ordinary http transfers that send a file from a single source in a continuous sequence from beginning to end, , BitTorrents are sent randomly from many sources and in a **rarest-first** scheme that first sends bits of a file that are harder to find before sending readily available bits.

**4** The universe of peers and trackers exchanging requests and bits of files is called a **swarm**. The chart from the BitTorrent client Azureus shows the ping times among more than a million members of a single swarm, all participating at the same time and all joined in the manner of "six degrees of Kevin Bacon." The trackers police the swarm, looking for **leeches**—persons who download files without contributing seeds of their own files. Trackers—or more precisely, the client software trackers use—can slow down sending bits to leeches and reward with faster download times seeders who contribute to the swarm.




**5** The second graph—Azureus' Giant Eye—shows one BitTorrent user's relationship to others in the swarm,represented by the circles orbiting him,with whom he is exchanging pieces of a hit movie that's only been in the theaters for a few days. The dark blue lines are his download connections and the light blue lines represent the uploads he's providing. Boxes travel along the lines at speeds proportional to the actual upload and download speeds. The gray circles in the orbit are computers that have been **snubbed** because their connections are so slow, they're a waster of time. In the center is a pie chart that represents how much of the movie that person's received, in this case about 75-percent.

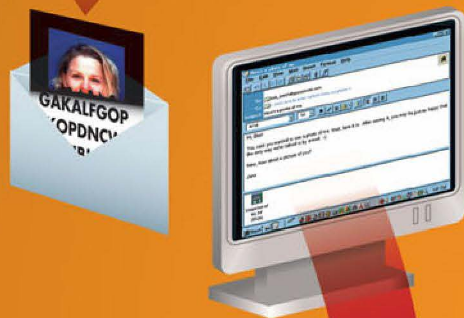


### Sure It's Fun, But Is It Legal?

Unless you live in Canada, where the supreme court ruled that people cannot be sued for downloading songs and movies, you're violating the law if you download copyrighted material, regardless of the methods. BitTorrent enthusiasts don't argue that point as much as they argue that BitTorrent has plenty of perfectly legal uses distributing public domain material and works created using common copywrite. And then, with its vagabond ways, BitTorrent makes it less likely for you to be nabbed with an illicit copy of the *Transformers* movie rolling onto your hard drive. BitTorrent transfers tend to be hit-and-run violations, making it harder to determine who's involved than it was in the Napster heyday when servers operated 24-7 with hard drives that brimmed over with goodies for the downloading. With the introduction of trackerless trackers, it becomes even harder to catch someone in the act. Not that it's impossible. The entertainment industry has successfully sued to close down BitTorrent sites that specialized in maintaining databases of other sites where songs and movies could be bittorn. At least one person, in Hong Kong, has been jailed. So the question is, are you feeling lucky? Well, are you?

# How Email Replaces Snail Mail

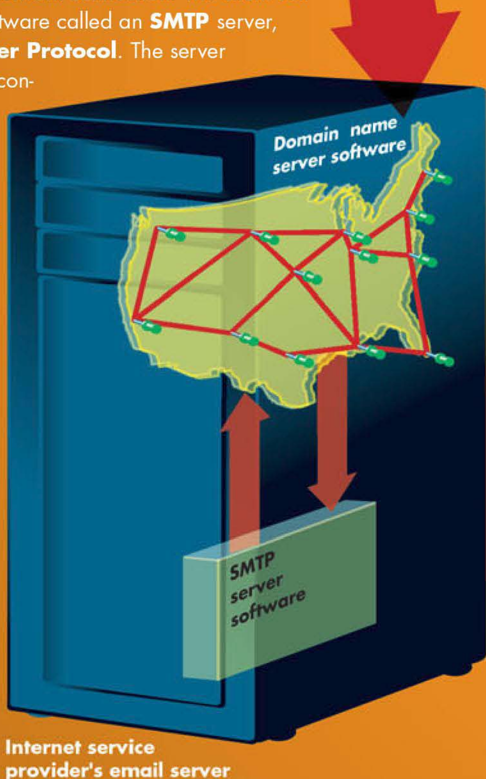
**1** Using email client software, Jane creates a message to go to Bob. She also attaches a digitized photograph of herself, which is encoded using a standard algorithm, such as MIME, uuencode, or BINHEX. Just as easily, Jane could enclose a word processing document, spreadsheet, or program.

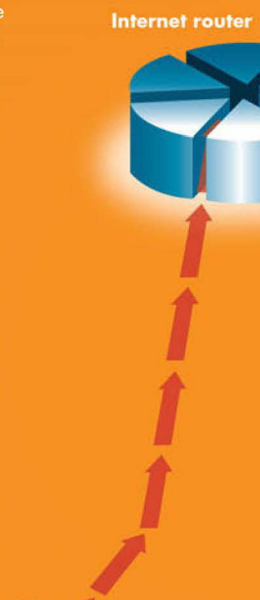


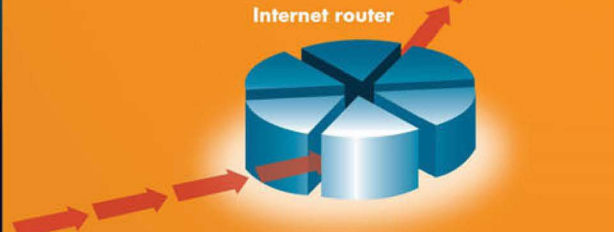
**2** The encoding turns the data making up the photograph into ASCII text, which computers commonly use for unformatted, simple text. The email software might also compress the enclosure before attaching it to the message so it takes less time to send.

**3** The client software contacts the Internet service provider's computer server over a modem or network connection. The client software connects to a piece of software called an **SMTP** server, short for **Simple Mail Transfer Protocol**. The server acknowledges that it has been contacted, and the client tells the server it has a message to be sent to a certain address. The SMTP replies with a message saying either, "Send it now," or "Too busy; send later."

**4** The client sends the message to the SMTP server and asks for confirmation. The server confirms that it has received the message.

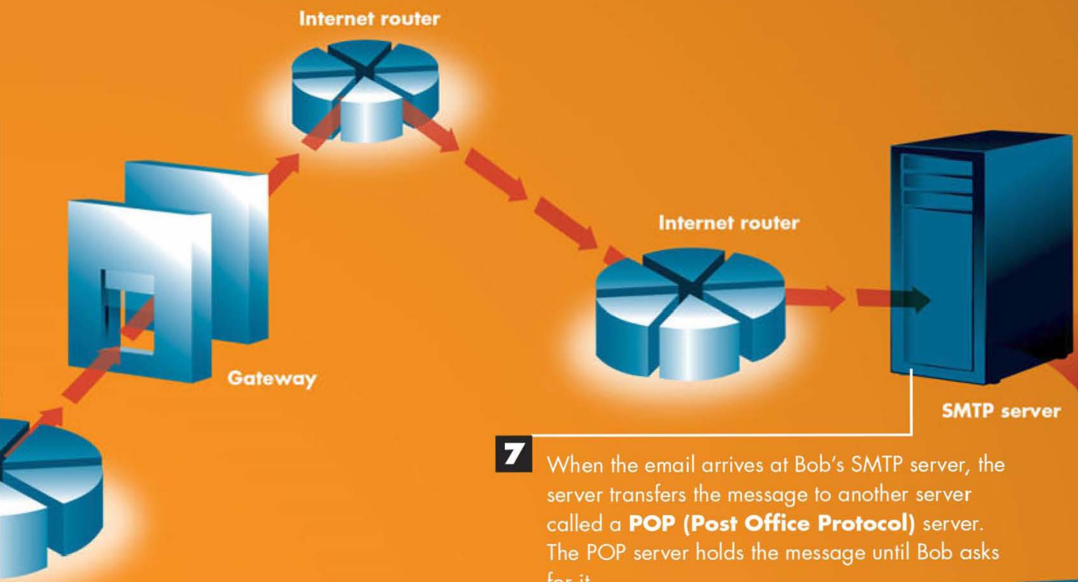


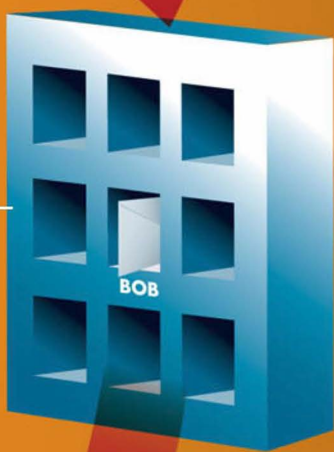




**5** The SMTP server asks another piece of software, a **domain name server**, how to route the message through the Internet. The domain name server looks up the domain name—the part of the address after the @ character—to locate the recipient's email server. The domain name server tells the SMTP the best path for the message.

**6** After the SMTP sends the message, the email travels through various Internet routers. Routers decide which electronic pathway to send the email along based on how busy the routes are. The message might also pass through one or more **gateways**, which translate the data from one type of computer system—such as Windows, Unix, and Macintosh—to the type of computer system that's the next pass-through point on the route.



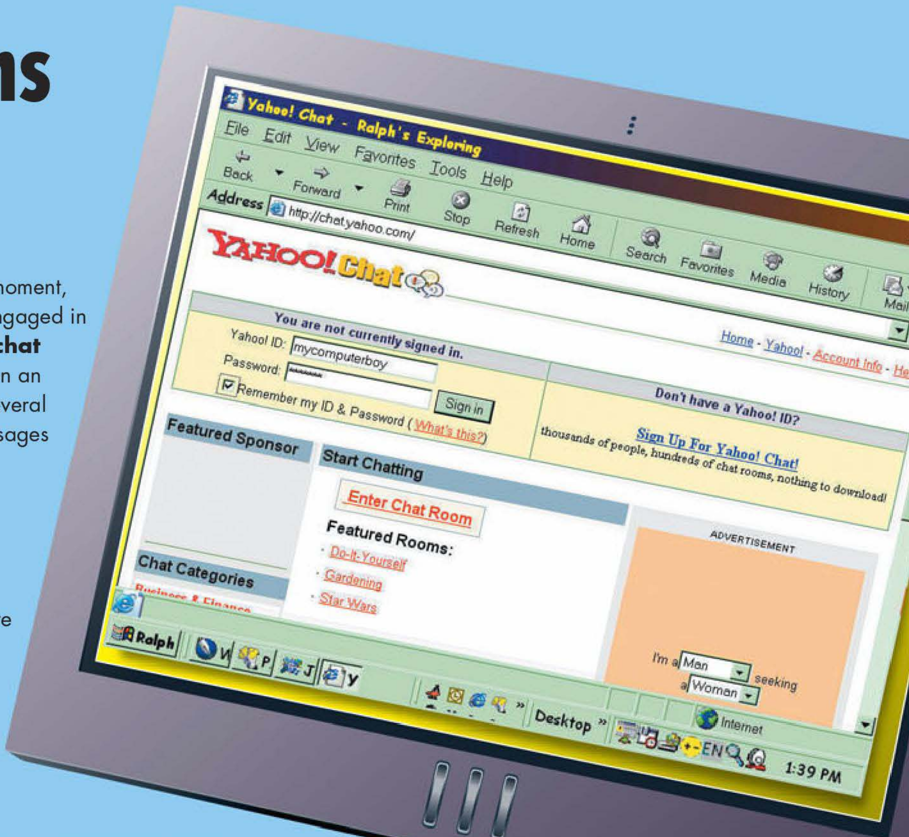

**7** When the email arrives at Bob's SMTP server, the server transfers the message to another server called a **POP (Post Office Protocol)** server. The POP server holds the message until Bob asks for it.




**8** Using his email client software, Bob logs on to the POP server with a username and password and then asks the server to check for mail.

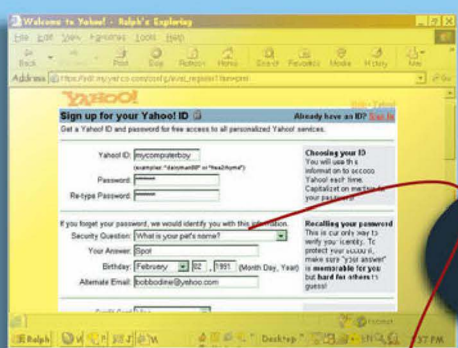
**9** The POP server retrieves Jane's stored message and transmits it to Bob's client software. Some email software decodes and decompresses any enclosures. Others would make Bob use a utility program to expand and decode the attachment. Either way, Bob can now read Jane's message and see what she looks like.

# How Chat Rooms Throw a Party

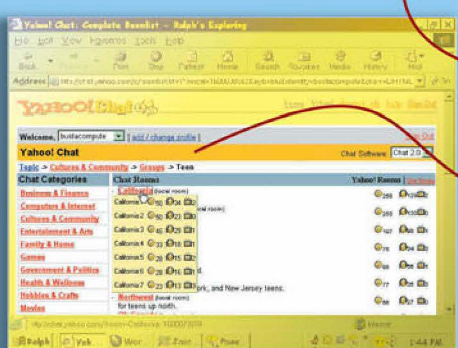



1 Computers make it hard to be a hermit. At this very moment, there are hundreds of thousands of computer users engaged in friendly **chats** over the Internet. Chats are held in a **chat room**, a virtual room that's really software running on an Internet server. The chat software is designed to let several computer users, all online at the same time, type messages that are seen simultaneously by all the other chatters.

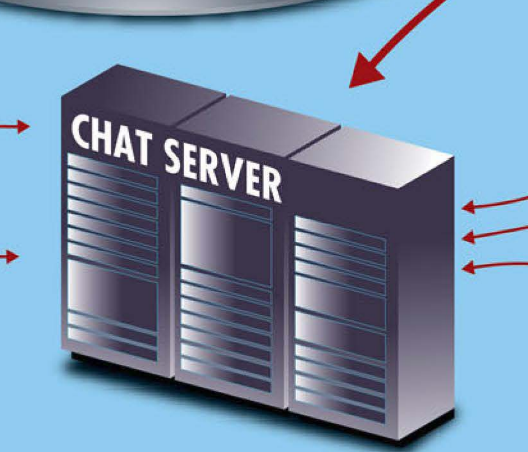
2 To join a chat, you must first run a chat **client** software on your computer. Client programs work over a network with a host program running on a hefty-sized server. The installation, performed by the server software, is barely noticeable. The chat client shown here is for Yahoo!, at chat.yahoo.com.

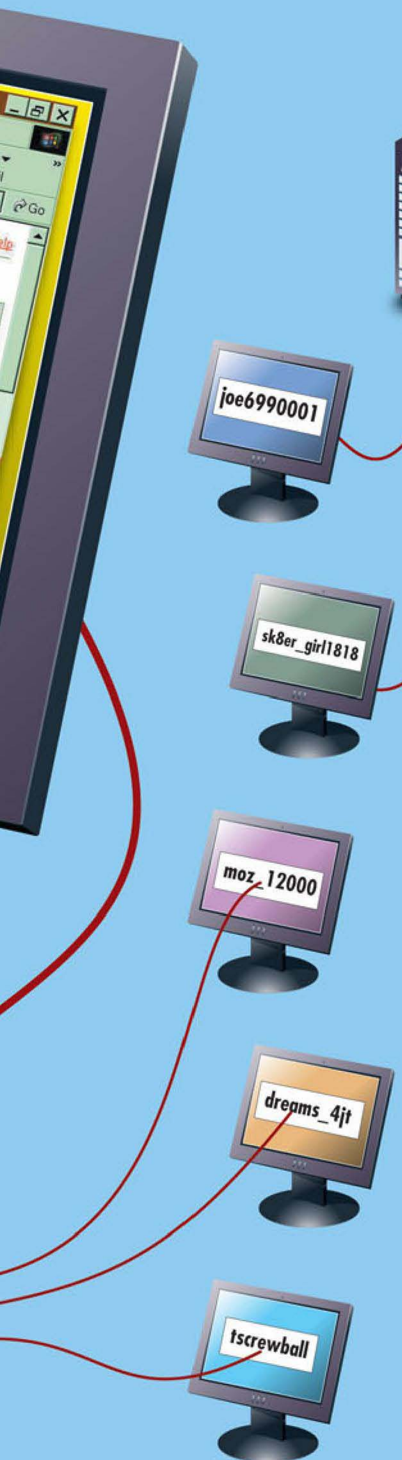



3 The first time you use a chat program, you set up a **screen name**, or **handle**, for yourself. Traditionally, chatters pick a name that's edgy. We'll go with Tabascorow.




4 After you log in under your screen name, you choose which chat room you want to enter. At the same time, thousands of other people are also signing on to the same server. Most of them head for a chat room devoted to a particular subject, such as computer games or politics.



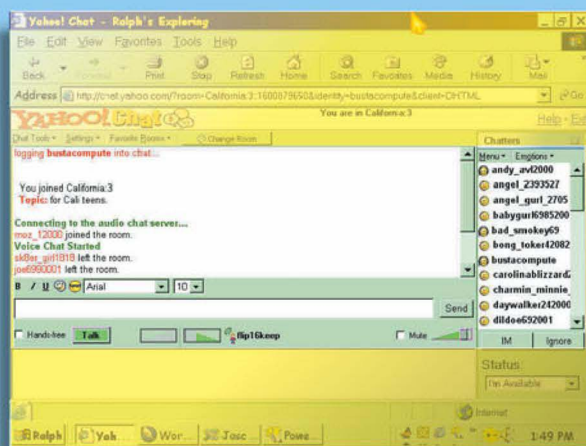




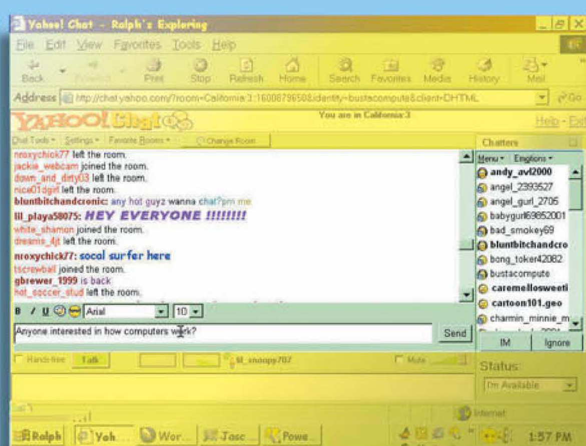
**5** When you make your choice, the chat server associates you with the path you took to it so that it will recognize any other messages from you. Then it adds your screen name to a list of other people who are already in the room.

**6** Finally, it sends a line of text that appears on the screens of everyone in the chat room announcing that Tabascorow has joined the room.

**7** Now anything you type is sent to the chat server. The server adds your screen name to the words you typed so others will know who's "talking," and sends them out to the computers of everyone in that chat room.




**8** They see everything you write, and you see everything everyone else writes. It can get confusing, especially in a crowded room. Most chat software lets you give your message a distinctive typeface, size and color that make it easier for everyone to follow conversation threads.



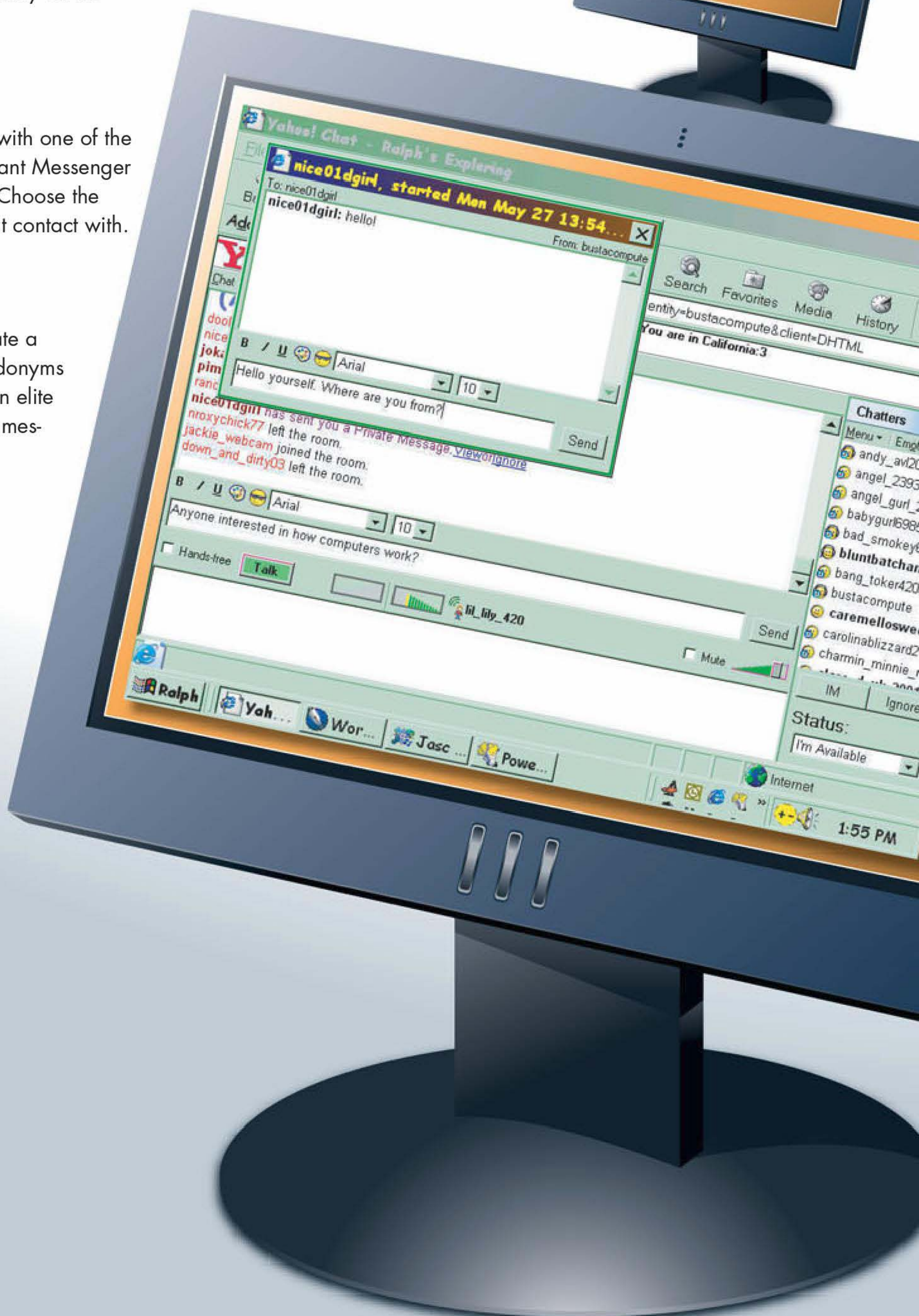
# How Instant Messaging Lets You Pass Notes

**1** While chat rooms are for public use, you need to use **instant messaging** software for a private conversation. It lets you and selected friends write messages that appear instantly on the screen of the person receiving it.

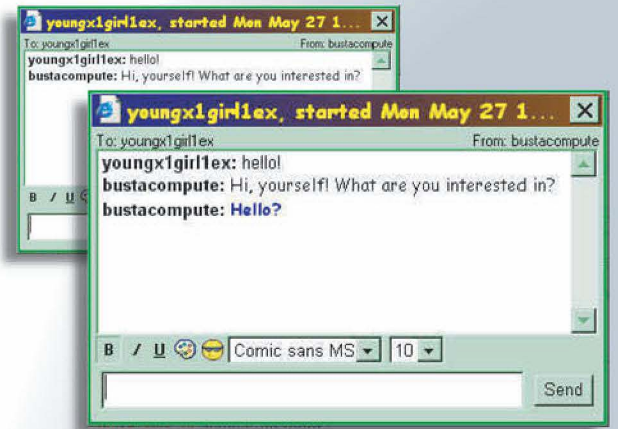
**2** To use Instant messaging, you must first sign up with one of the free instant messaging service, such as AOL Instant Messenger (AIM), MSN Messenger, or Yahoo! Messenger. Choose the one used by the people you want to be in instant contact with.

**3** With the messaging service's software, you create a **buddy list** of your contacts, using online pseudonyms called **handles**. The list is like the bouncer at an elite night club. If a name isn't on your buddy list, its messages never appear on your PC.


**4** Once installed, whenever you log onto the message service, the software running on your computer sends both your handle and buddy list to a central computer, a **messaging server** that takes care of thousands of users simultaneously.

**5** The message server adds your name to a list of people who are online. It also compares your buddy list with the handles of people online, and sends a message to let you know who's available for instant messaging.






**6** When you pick a name from that list, a small window opens up where you can type a short message. When you click on send, the message is sent to the messaging server, which passes it along to your buddy.

**7** On the buddy's computer screen, a window opens to display what you wrote. The buddy can type a quick reply to you.

**Hands-on**

AOL instant messaging can't exchange notes with Yahoo! messaging or with MSN. If you use AOL but a friend uses MSN, the two of you ordinarily can't swap messages. However, programs such as Trillian work with any of the major messaging services. You still have to register with your friend's service, but you communicate with all your buddies, and Trillian does the necessary translations to make it seem as though you were all using the same service.

CHAPTER

# 28 How Wireless Sets PCs Free

**WIRELESS** was once a quaint term we encountered in British WWII movies in which Britons would huddle around massive radios. Only they called it the "wireless" because the leading-edge technology of radios was that, unlike telephones and telegraphs of its day, a radio received music, news, and entertainment without being connected to anything but an electrical outlet.

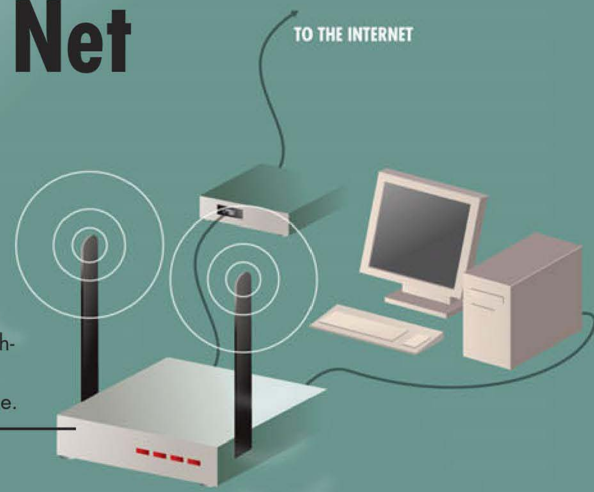
Today, **wireless** has a new meaning. Radio's offspring, television, has become increasingly wired as broadcasters learn how to pump more broadcasting over a cable connection than they could over the airwaves. For a while—maybe a couple of years in the 90s—someone who was in the in crowd technologically was said to be "wired." The more wires—cable, DSL, T1 Internet connections, piped-in music channels, voice lines, fax lines—you had leading to your desk was a sign of how connected you were to the world's hustle and bustle.

Forget it. Now, having a jumble of wires creating some electrical monster's nest beneath your desk is the mark of someone who is no longer on the bleeding edge. The freedom of wirelessness has become an important issue as people are increasingly on the go and can't be tied to a desk or even a single address. Even if you were willing to put up with the immobility, the rabbit-like multiplication of devices that need to swap information with each other is increasing to the point that stretching wires among them all would be impossible.

Today's technology makes possible wireless devices that were impossible in the old wireless era. Radio signals back then had to use a relatively large amount of the spectrum of radio waves that could carry a signal a long distance. Now, engineers have devised ways of packing more information into the same bandwidth and ways to let several devices use the same airwaves at the same time without jumbling everything. New, too, are the devices that don't require the power to broadcast over vast distances, as radio and TV originally had to do. Today, your laptop needs to talk to that printer in the corner. Your cell phone and pagers are, at least in the cities, within range of smaller, less encompassing radio-wave transmitters and receivers. And wirelessness is coming to devices we didn't conceive of a decade ago: Refrigerators wirelessly send automatic grocery lists to PCs; keyboard, mice, and game controllers are shedding their tethers. WiFi radio connections are creating **hot spots** in cities around the world where anyone with a laptop can wirelessly plug into the Internet. Look at anything with wires running out of it; Soon—probably within the next decade—those wires will be as quaint and antique as a telegraph key.

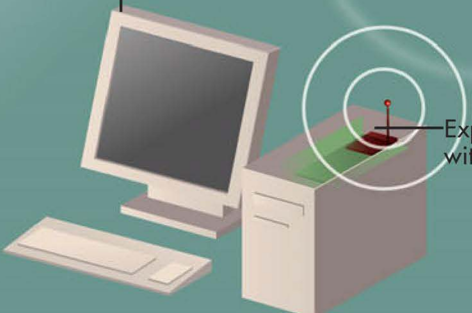
In this chapter, we'll lead off with the device that started the current wireless mania—the cell phone—and how WiFi and cell phones are now giving us ubiquitous access to the Internet. We'll also touch on how a technology with the unlikely name of Bluetooth is going to make devices in the home and office smarter and more communicative, with you and with each other. And it's all done without wires.

# How Wi-Fi Spreads the Net Everywhere




**1** Any of the varieties of IEEE 802.11 wireless networks begins with an **access point**, or **AP**, a network node connected directly to a wired local area network or to the Internet. An AP also includes a radio transmitter and receiver operating in either the 2.4GHz radio band or the 5GHz range. (See box for the distinguishing characteristics of the three popular 802.11 flavors.) The access point might also provide ordinary RJ-45 Ethernet jacks for connecting nearby nodes via cable.

**2** The other wireless nodes, called **stations**, each have a transmitter/receiver working in a matching bandwidth. A desktop PC uses an antenna-equipped expansion card or a USB antenna attachment. Notebook computers use a PC Card, USB attachment, or have the radio components built into the notebook.




**4** If an access point within that range picks up the probe request, the AP broadcasts an acknowledgment, and the two go through whatever security or payment arrangements the network has set up. They also establish any settings needed so the notebook can talk with the rest of the network. If two APs reply to the probe, the notebook will use whichever has the stronger signal.

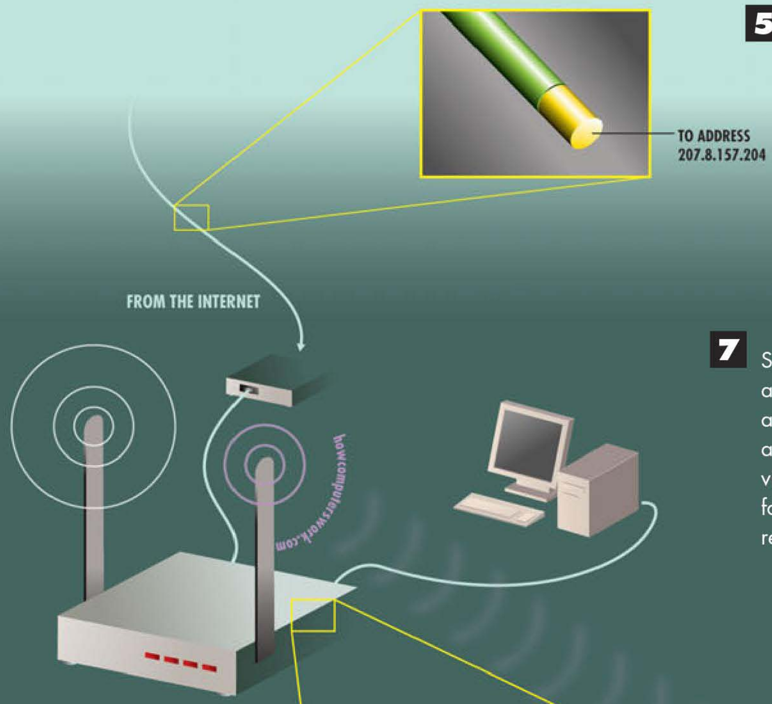


**3** When a notebook, for example, connects to the network, it broadcasts a **probe request** identifying itself and asking if any other 802.11 devices are within **range**. That range covers a 25-foot to 150-foot radius. The range is a factor of the frequencies, transmitter signal strengths, and the sensitivity of the receivers. Lower frequencies travel farther than higher frequencies.

| STANDARD | 802.11b | 802.1a | 802.11g |
|---|---|---|---|
| SPEED | 11 Mbps | 54 Mbps | 54 Mbps |
| RANGE | 100-150 feet indoors | 25-75 feet indoors | 100-150 feet indoors |
| FREQUENCY | 2.4GHz, a band already crowded with cordless phones | 5GHz, an uncrowded band. | 2.4GHz, still a crowded of cordless phones and microwaves |
| ACCEPTANCE | Hot spots are already established using 802.11b. Equipment is readily available. | More common in corporate and office environments. | 802.11g is compatible with the specs for 802.11b, meaning it can be used on a network based on b or g versions. |

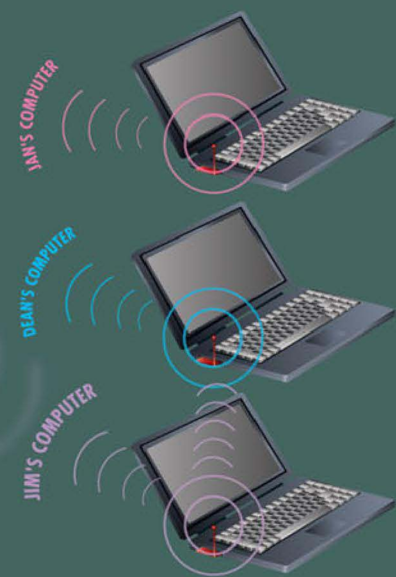
### A, B, or G?

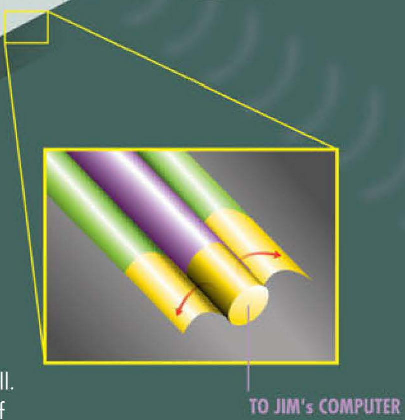
Currently there are three forms of the 802.11 standard proposed by the IEEE (Institute of Electrical and Electronics Engineers) for wireless networking: 802.11b, which came, strangely enough, before 802.11a, with the next proposal skipping to 802.11g. Each standard has advantages, but 802.11g has the speed, compatibility, and range to replace 802.11b as the most common configuration of Wi-Fi. Above are their pros and cons.



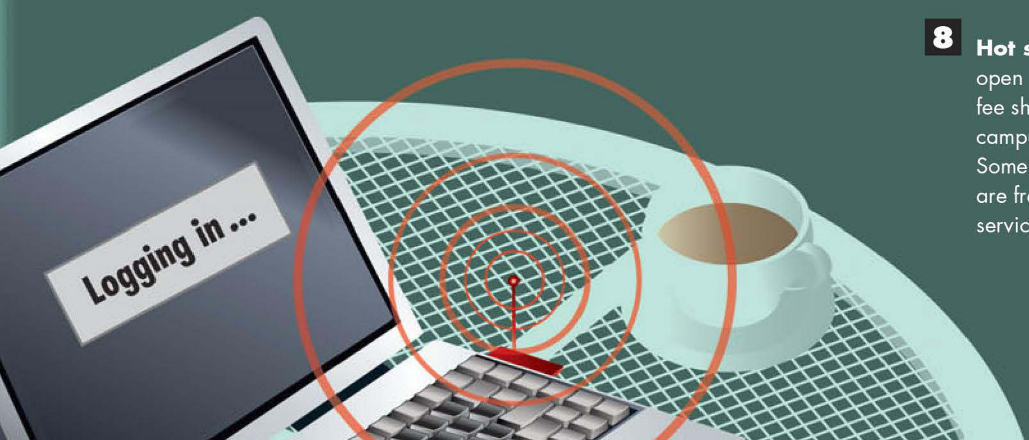

**5** Many access points are also routers, dividing Internet access among several stations on the **wireless local area network (WLAN)**. The Internet sees the entire WLAN as a single Internet address, such as 207.8.157.204. It's called an **IP (Internet protocol)** address, and it's how computers see the Internet addresses that we see as words, such as www.howcomputerswork.net.

**7** Stations are not limited to communicating only through the access point. Stations can exchange information with one another, unaided, on a **point-to-point** basis. The network as a whole can extend beyond the range of the routers, provided there are occasional **extension points** that pick up fading communications to and from access points and rebroadcast the signals with renewed strength.




**6** When the router receives a packet from the Internet, it strips off an encrypted and addressed outer shell. Inside is the name of the real recipient of the packet, one of the stations on the WLAN. The router checks the address against a list of rules set up by the network administrator. Unless the rules forbid it, the router passes the packet to the destination within the network. For outgoing messages, the router shrouds each packet in a shell that hides the packet's true origin and gives only the router's ID as a return address.




**8** **Hot spots** are areas where 802.11 networks are open to the public. The first have appeared in coffee shops, and they are a growing presence on campuses, at airports, hotels, and office lobbies. Some hot spots charge for Internet access; others are free as a marketing ploy or as a public service.

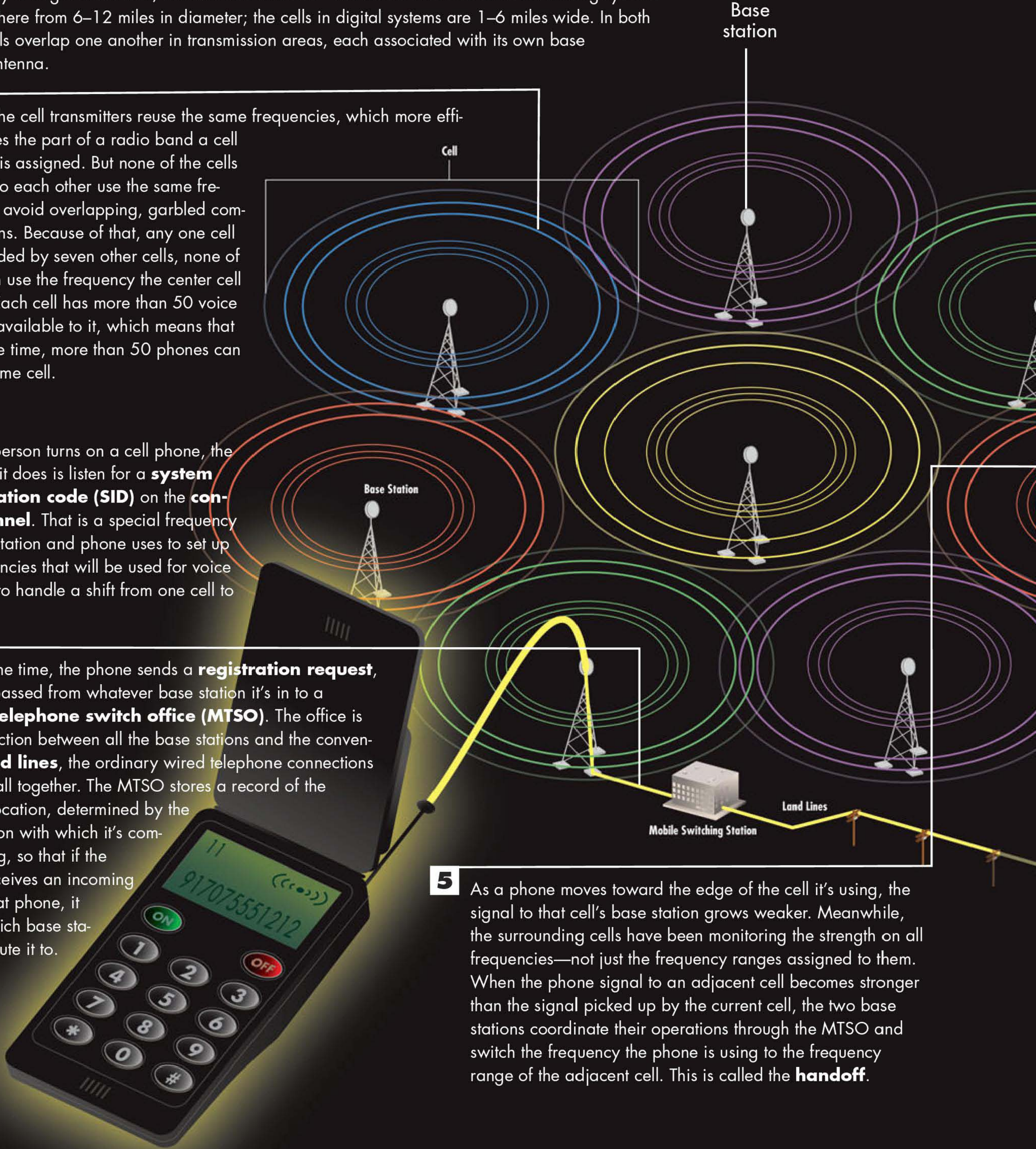
# How Cell Phones Make a Call

**1** Cellular phones—whether **analog** or **digital**—derive their name from **cell**, which means the area covered by a single transmitter/receiver and antenna—the **base station**. Cells for analog systems are anywhere from 6–12 miles in diameter; the cells in digital systems are 1–6 miles wide. In both cases, cells overlap one another in transmission areas, each associated with its own base station/antenna.

**2** Many of the cell transmitters reuse the same frequencies, which more efficiently uses the part of a radio band a cell company is assigned. But none of the cells adjacent to each other use the same frequency to avoid overlapping, garbled communications. Because of that, any one cell is surrounded by seven other cells, none of which can use the frequency the center cell is using. Each cell has more than 50 voice channels available to it, which means that at any one time, more than 50 phones can use the same cell.

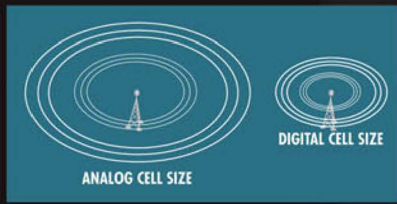
**3** When a person turns on a cell phone, the first thing it does is listen for a **system identification code (SID)** on the **control channel**. That is a special frequency the base station and phone uses to set up the frequencies that will be used for voice and how to handle a shift from one cell to another.

**4** At the same time, the phone sends a **registration request**, which is passed from whatever base station it's in to a **mobile telephone switch office (MTSO)**. The office is the connection between all the base stations and the conventional **land lines**, the ordinary wired telephone connections that tie it all together. The MTSO stores a record of the phone's location, determined by the base station with which it's communicating, so that if the MTSO receives an incoming call for that phone, it knows which base stations to route it to.




**5** As a phone moves toward the edge of the cell it's using, the signal to that cell's base station grows weaker. Meanwhile, the surrounding cells have been monitoring the strength on all frequencies—not just the frequency ranges assigned to them. When the phone signal to an adjacent cell becomes stronger than the signal picked up by the current cell, the two base stations coordinate their operations through the MTSO and switch the frequency the phone is using to the frequency range of the adjacent cell. This is called the **handoff**.

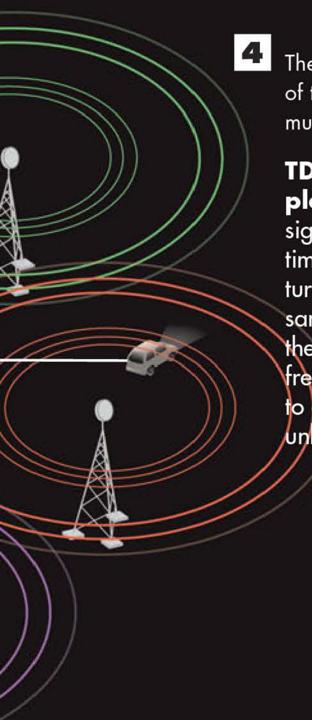
# Analog Versus Digital




## Analog

**1** In an analog cell system, the base station uses a method called **frequency division multiple access (FDMA)**, in which the MTSO assigns **duplex channels**—two different carrier frequencies within the range assigned to the base station. This allows the base station and the phone to transmit signals to each other simultaneously. Typically, each analog system has 95 duplex channels. Each carrier uses about 30KHz of bandwidth within the 800MHz range of radio signals.

**2** The analog carrier signals are modified by the sound pattern created by the conversations they carry. When the signals are received at either end of the connection, the carrier signal is stripped away, leaving only the analog pattern of the sound it was carrying. In turn, that signal is amplified so that it re-creates the sound through the phone's earpiece.



## Digital

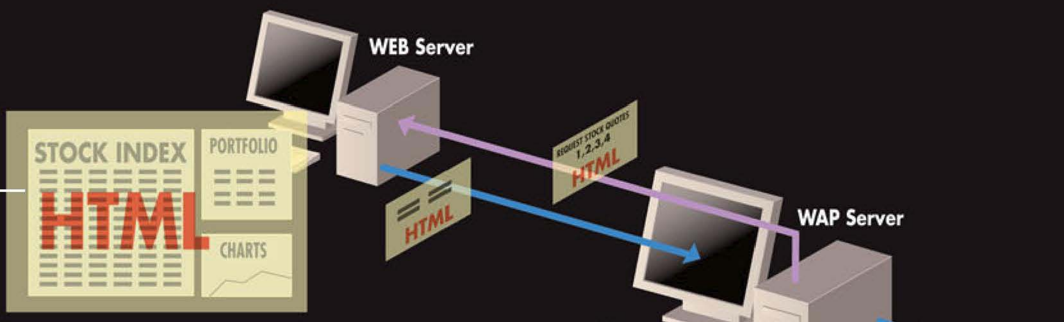
**3** Digital cell systems convert all conversations and other sounds from an analog form to a binary pattern of 0s and 1s that is then compressed. This allows digital systems to carry more conversations than analog can at the same time. And because the signals are digital, they can be used to transmit data, such as written messages and computer files, in addition to voice signals.

**4** The digital system uses one of two methods to handle multiple connections:

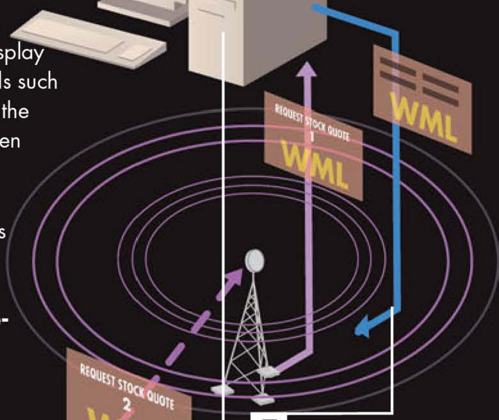
**TDMA (Time division multiple access)** divides the carrier signals into many small slices of time. Different conversations take turns using time slices on the same radio frequency. Because the time slices rotate use of the frequency so rapidly, it appears to each cell user as if it were one unbroken signal.

**CDMA (Code division multiple access**, also called **spread spectrum)**assigns a code to each packet of digital data being sent as part of a single conversation. The packets are spread among the available frequencies, all of which are monitored constantly by the phone and base station. Receivers on both ends of a single conversation pick up packets from all conversations going on at the same time. The receivers use the packet codes to identify packets that are part of the same conversation and reassemble the individual packets into separate, unbroken signals.

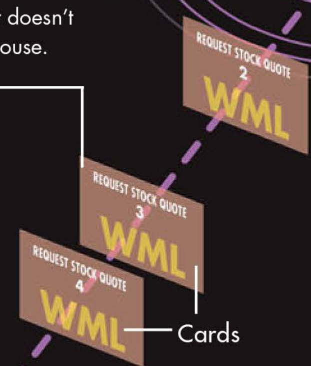
# How Wireless Internet Keeps You Connected



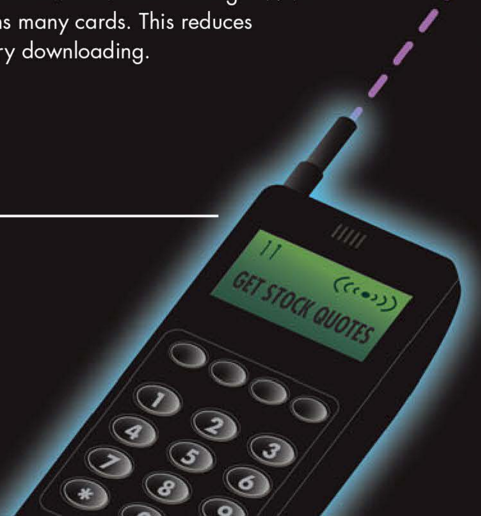

**1** Ordinarily, when you request information from the Internet, you receive a document formatted with the **hypertext markup language (HTML)** (see Chapter 30). HTML is designed to display information on large screens over a connection fast enough to include frills such as graphics and animation. In addition, HTML doesn't understand any of the information it's formatting; it doesn't know what letter a character is or even whether it's a character, number, or punctuation.




**2** A cell phone, pager, or palm PC designed to connect to the Internet uses **wireless application protocol (WAP)**. WAP is a set of standard instructions designed specifically to securely transmit information for displaying on tiny screens over costly cell connections. WAP uses the **wireless markup language (WML)**. It's similar to HTML but doesn't assume the receiving device has a qwerty keyboard or a mouse.



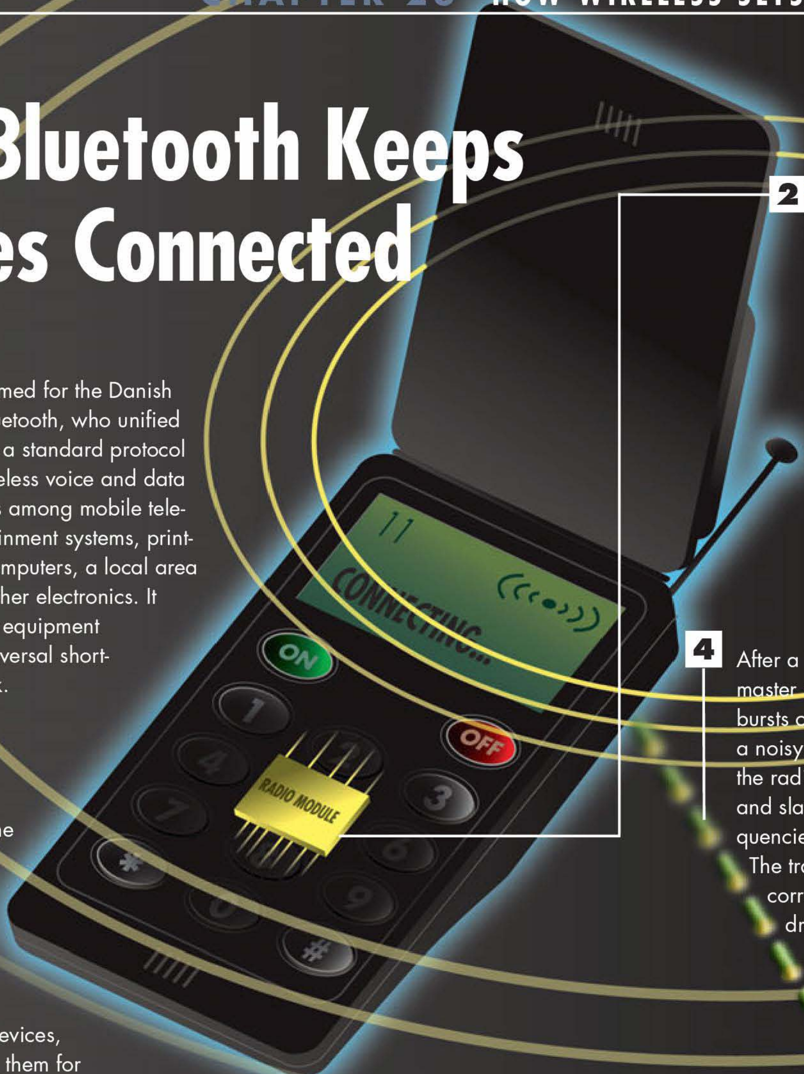

**3** When you send a request for information through a wireless device, WAP doesn't bother to send the entire message. Instead, it sends one of several **cards** available to it. Each card is a separately displayed screen. The cards work with the commands to connect to specific sites and retrieve, say, a stock quote or address map, depending on the input from the user. Unlike with plain HTML—where every link you click on downloads a new HTML file—a single WML file contains many cards. This reduces unnecessary downloading.




**4** The request goes to a computer server acting as a WAP gateway. From the type of card and encoding, the gateway translates the card into ordinary HTML and relays the request to the destination server on the Internet.

**5** That server sends its response to the WAP gateway, which extracts the data and encodes the file into **bytecode**, which is then transmitted to the WAP browser. Bytecode is a source code—generalized computer instructions that the WAP browser compiles to fit the specific needs of whatever type of device (phone, PDA, pager) on which it's running. Finally, the browser uses WML to display the information.

# How Bluetooth Keeps Devices Connected




**1** **Bluetooth**, named for the Danish King Harald Bluetooth, who unified Scandinavia, is a standard protocol for unifying wireless voice and data communications among mobile telephones, entertainment systems, printers, portable computers, a local area network, and other electronics. It connects all the equipment through one universal short-range radio link.

**2** The standard is incorporated in a **radio module**, a microchip that can send both voice and data signals for about 30 feet, or 10 meters (100 meters with a power amplifier). Its signals operate in the free 2.45GHz ISM (**Industry, Science and Medicine**) band which non-Bluetooth devices also use.

**3** In command of the Bluetooth protocol is each device's **link manager (LM)**. This software identifies other Bluetooth devices, creates links with them for voice or data, and sends and receives data at a theoretical 1Mbps (725Kbps, real world). The link manager also determines the mode in which Bluetooth operates:

**4** After a link is established between master and slave, Bluetooth sends short bursts of data in packets. To survive in a noisy radio frequency environment, the radio transmissions from master and slave hop among 79 different frequencies to send each packet of data. The transmission also includes error correction in case a packet is dropped or garbled.


**Standby, or sniff, mode:** An unconnected unit periodically listens for messages every 1.28 seconds. Each time a device wakes up, it listens on a set of 32 hop frequencies used to address that unit.

**Page mode:** A Bluetooth device makes itself a **master** by initiating a link to another device, the **slave**. The master tries to find the slave by broadcasting an identical page signal on 16 different hop frequencies defined for the device being sought.

**Inquiry mode:** If the page fails or if the master doesn't know what other devices are available, as in the case with public printers and fax machines, the master transmits an inquiry on the remaining hop frequencies. This causes receiving devices to identify themselves, and the master then sends a page specific to one of the newly discovered devices. The maximum delay to establish a link is about three seconds.

**Park mode:** A device wakes up at regular intervals to listen to the link, synchronize with the rest of the devices, and check for page messages.

**Hold mode:** When a device is turned off to save power, any other Bluetooth device can wake it up.

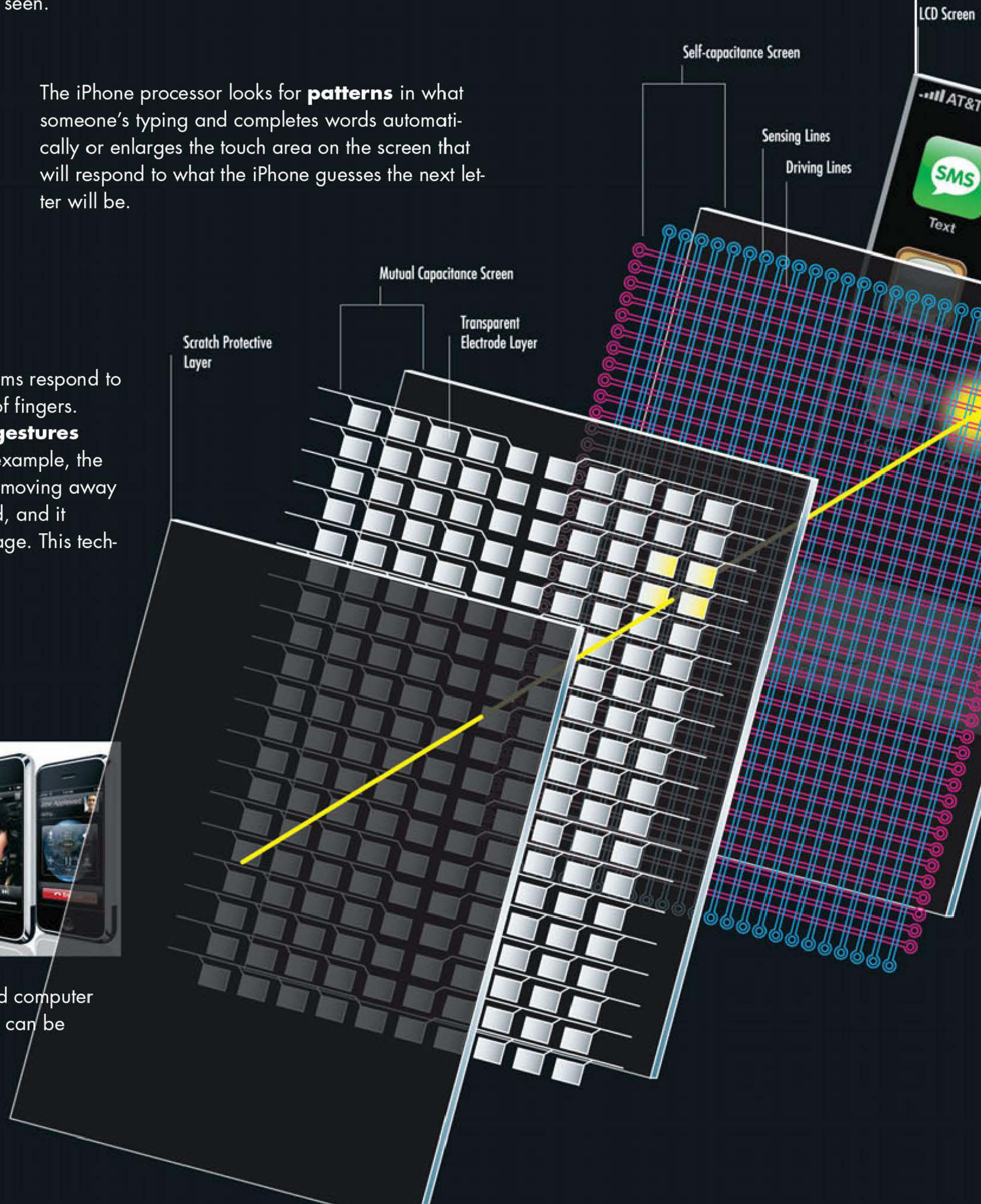
# How the iPhone Makes It All Slick

There's not a lot the Apple iPhone does that hasn't been done before by other wireless phones. Internet surfing, snapping photos, playing MP3s, maintaining contact lists, and, of course, making phone calls are all done by other phones, too. In fact, there are few nattering complaints about things the iPhone doesn't do as well as other cell phones, such as the way it makes you search for someone in its phone book. None of that matters. The difference between the iPhone and all other cell phones—and it's a difference that commands a $400 list price—is that the iPhone does it all slicker. That, and the fact that the iPhone crossed the lines dividing cell phones and computers and video, making it the smallest, most desired portable computer anyone's ever seen.

A flexible printed circuit connects the circuit board to the 3.5 inch 480×320 pixel screen/keyboard.

The iPhone processor looks for **patterns** in what someone's typing and completes words automatically or enlarges the touch area on the screen that will respond to what the iPhone guesses the next letter will be.

The phone's complex touch mechanisms respond to the pressure, shape, and movement of fingers. They allow the screen to respond to **gestures** made with one or more fingers. For example, the iPhone interprets a finger and thumb moving away from each other as a zoom command, and it responds by enlarging the screen image. This technology is called **multitouch**.




The iPhone **multitasks** just as a full-sized computer does. While you surf the Web, the phone can be updating your phone contacts.


Infrared LED and sensor detect when the phone is removed from the ear and hangs up.

Jacks for earphones and power cord.

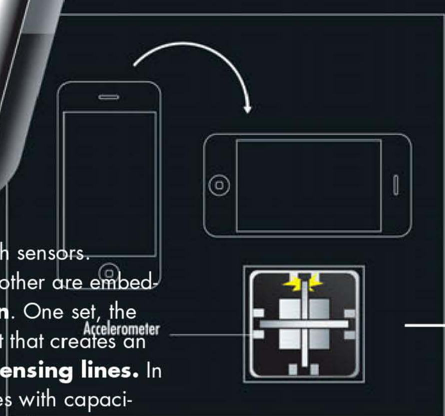
SIM card for phone service activation

Circuit board with two micro processors

Battery

Antennas for cellular service, bluetooth, and WiFi.

2-megapixel camera.

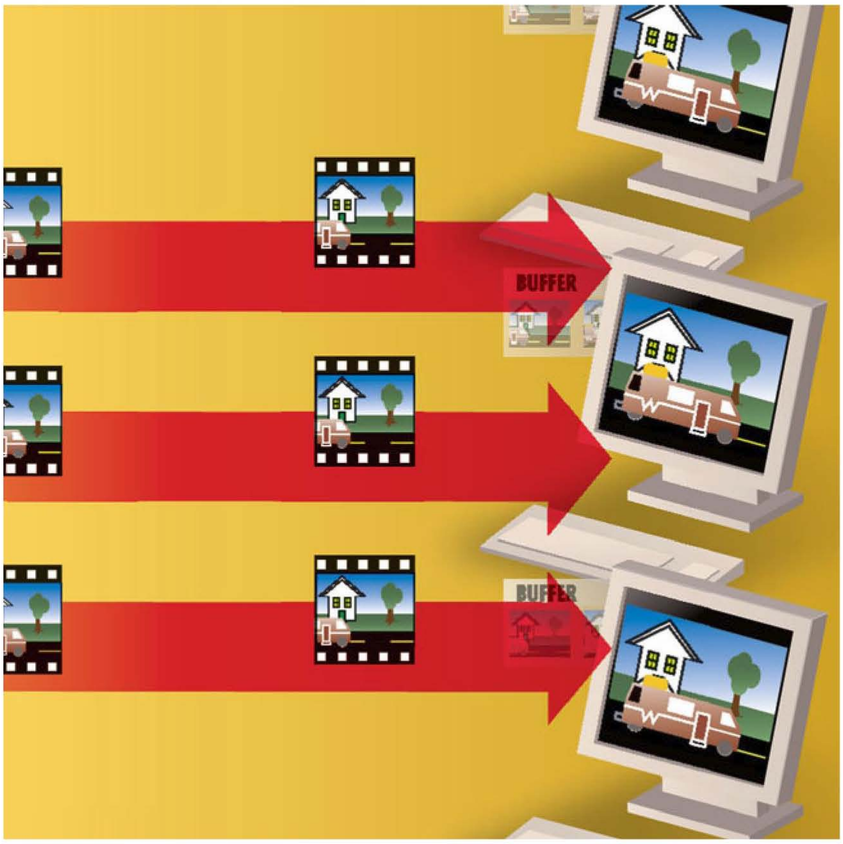



An **accelerometer** detects whether the iPhone is being held vertically or horizontally and changes the screen display to match the phone's orientation.

The screen incorporates two types of touch sensors. Electrical lines set at right angles to each other are embedded in the **mutual capacitance screen**. One set, the **driving lines**, carry an electrical current that creates an electromagnetic field that is read by the **sensing lines.** In the **self-capacitance screen**, electrodes with capacitance-sensing circuits are buried throughout the screen. When a finger touches the iPhone's screen, it creates a change in the electromagnetic fields that is detected by both types of senses. There, data is sent to the phone's processor, which interprets it in the context of what's on the screen and what application is running.

CHAPTER 29

# How the Net Provides Video and Audio on Demand



**TODAY** you use the World Wide Web to listen to Mozart and watch Super Bowl highlights, but the Internet began as a text-only medium. In fact, the Internet was a multimedia late-bloomer compared to personal computers. That's because the problem of how to handle the necessarily huge amounts of data involved in graphics, audio, and videos was easier to solve on the desktop than on the Internet.

Conveniently, there's a name for the problem: **bandwidth**. The term is used to describe how much data you can push across a network, a computer bus, or any of the many other data pathways that let components work with the same information. The wider the bandwidth, the better.

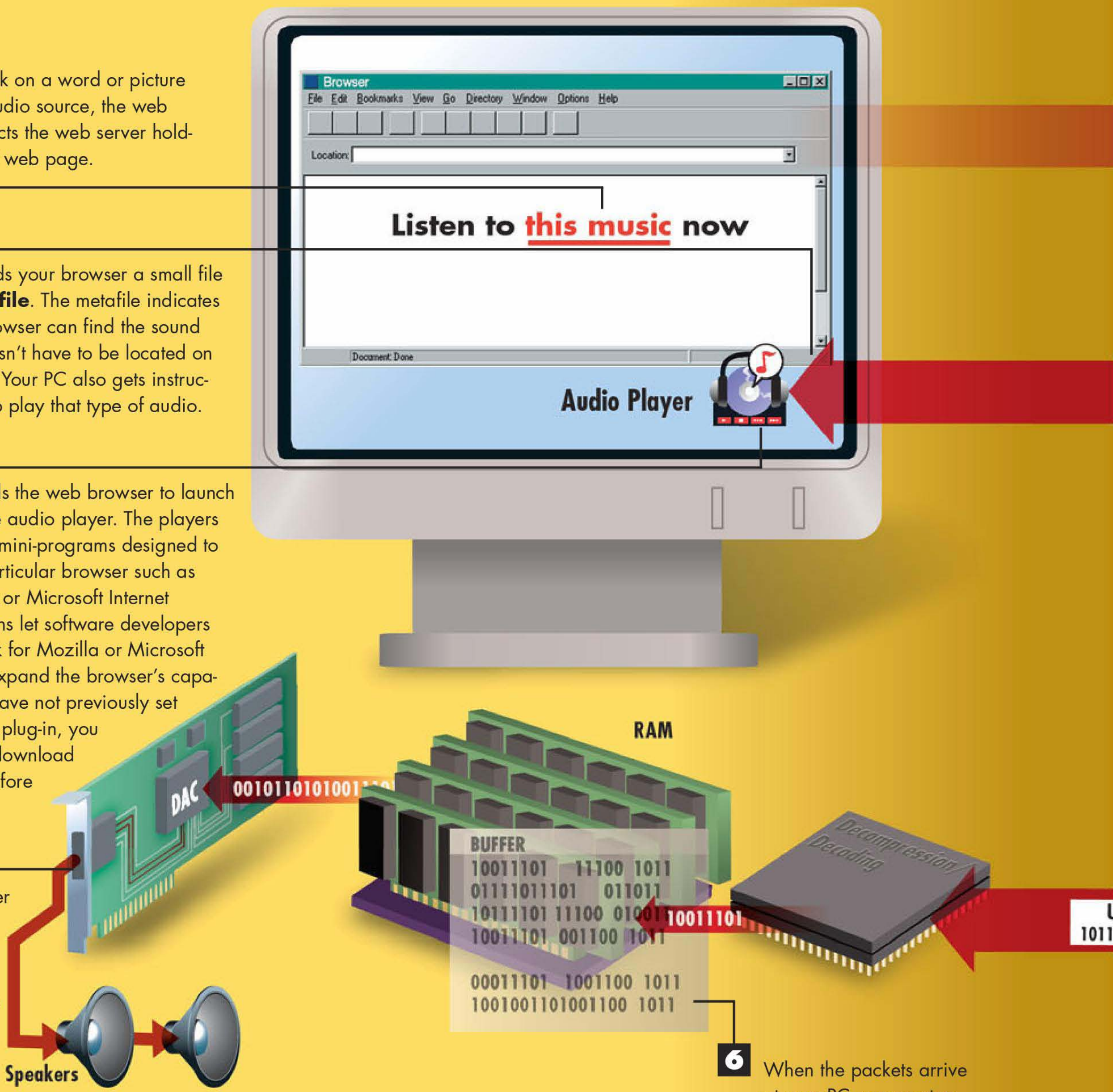
The Internet's bandwidth is growing at rates unimagined only a few years ago. The move toward cable and DSL as the connections of choice to the Net means that the Internet is losing a reputation for jerky, small video and tin-can audio. The tricks used today to overcome the narrow bandwidth of traditional modems, described in this chapter, will soon be only of historical interest. If your home doesn't yet have multiple PCs, it probably will eventually, and at least one of them is destined to become your television, radio, stereo, and telephone—it'll blend into one universal source of information and communication.

The first audio and video on the Internet were short clips because a computer had to download them completely to its hard drive before it could play them. A newer technology called **streaming** extends the length of a multimedia clip from seconds to hours. Streaming, used with a variety of players and audio/video formats, enables your PC to play the file as soon as the first bytes arrive, instead of forcing the PC to wait for an entire multimedia file to finish downloading.

Streaming doesn't send files as other files are sent on the Internet; that is, it doesn't use the same protocol. A **protocol** is the rules governing how two computers connect to each other—how they break up data into packets and synchronize sending them back and forth. Instead of the **Transmission Control Protocol (TCP)** used to track most Internet transactions, streaming calls on the **User Database Protocol (UDP)**. The crucial difference between the two protocols is how they check for errors. If you're downloading a hot new game off the Internet and a passing electrical interference garbles one packet, TCP suspends the download while it asks the sending computer to resend the bad packet. But with audio and video, if you miss a frame or word here or there, the loss isn't crucial. You might not even notice it. But you would notice if the protocol took the extra time to enforce the retransmission. That's why UDP lets the connection lose occasional packets without fussing.

Because of streaming technology, you can listen to a live concert located across the country, select from an archive of interviews with celebrities and scholars, or take a video call from beyond an ocean. Streaming technology proves that audio and video are more than fancy ways to dress up a Web page. They add immediacy to Internet communications.

# How a PC Plays Streaming Audio



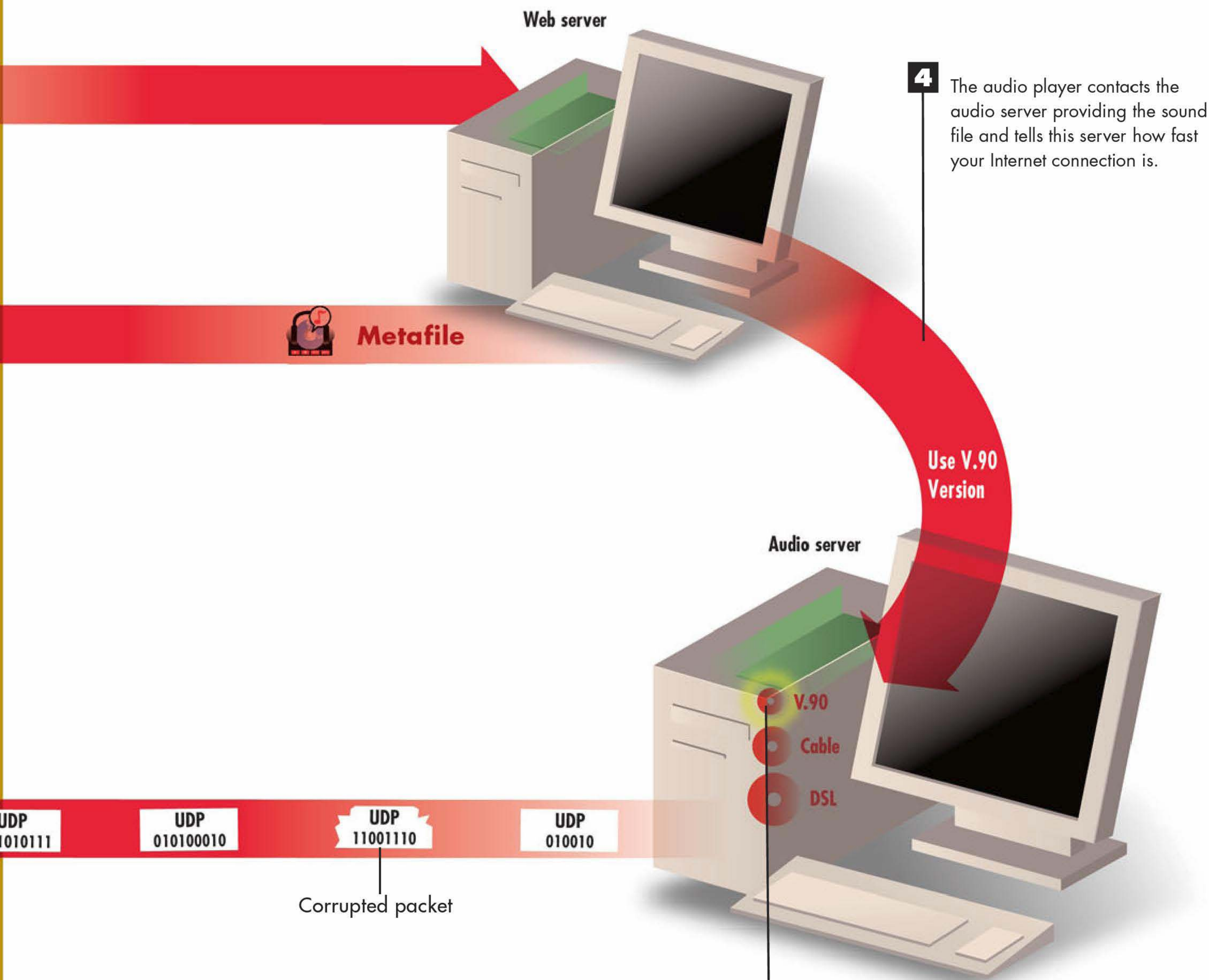

**1** When you click on a word or picture linked to an audio source, the web browser contacts the web server holding the current web page.

**2** The server sends your browser a small file called a **metafile**. The metafile indicates where your browser can find the sound file, which doesn't have to be located on the first server. Your PC also gets instructions on how to play that type of audio.

**3** The metafile tells the web browser to launch the appropriate audio player. The players are **plug-ins**, mini-programs designed to work with a particular browser such as Mozilla Firefox or Microsoft Internet Explorer. Plug-ins let software developers who don't work for Mozilla or Microsoft write code to expand the browser's capabilities. If you have not previously set up a particular plug-in, you might have to download and install it before continuing.

**6** When the packets arrive at your PC, your system decompresses and decodes them and sends the results to a **buffer**, a small portion of RAM that holds a few seconds of sound.

**7** When the buffer fills up, the audio player starts to process the file through your sound card, turning file data into voices, music, and sounds while the server continues to send the rest of the audio file. This process can continue for several hours. The buffer can temporarily empty if it doesn't receive enough data to replenish it. This happens if you access another web page, if you have a poor connection, or if Internet traffic is heavy. If the buffer empties, the audio replay pauses for a few seconds until your PC accumulates enough data to resume playing. If the sound source is live, the player will skip portions of the audio program. If the sound source is prerecorded, the player will continue from the point it stopped.




**4** The audio player contacts the audio server providing the sound file and tells this server how fast your Internet connection is.

**5** Based on the speed of your connection, the audio server chooses one of several versions of the audio file. It sends higher-quality sound, which requires a wider bandwidth, over faster links, and lower-quality sound over slower connections. The server sends the audio files to the PC as a series of packets in **User Database Protocol (UDP)**, which permits the occasional packet to get lost without critically disrupting the transmission.

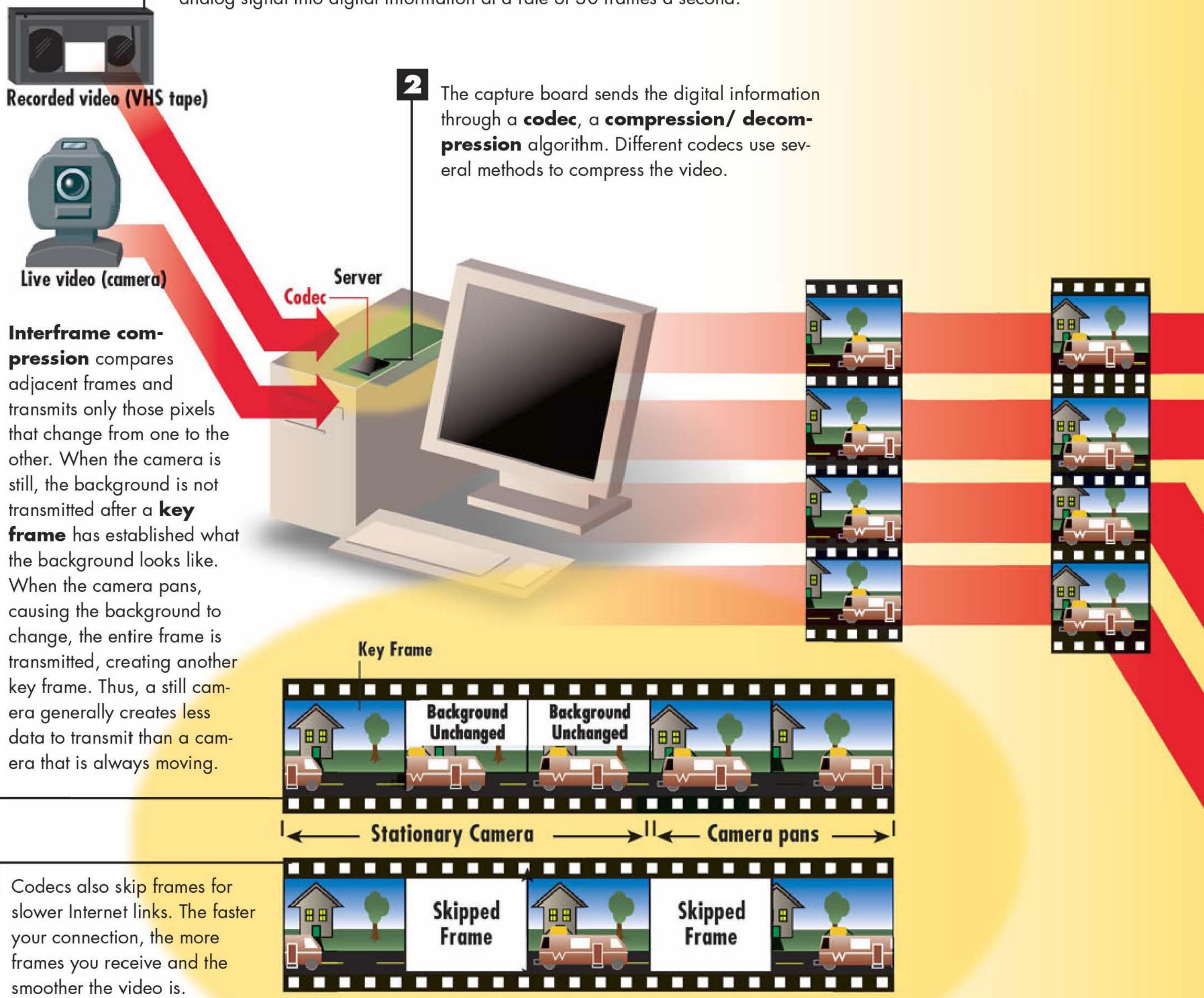
# How a PC Displays Streaming Video

**1** Inside the computer that acts as a server for Internet video, a video capture expansion card receives the ordinary analog video signal from its source, either a live feed or recorded tape. The capture board turns the analog signal into digital information at a rate of 30 frames a second.

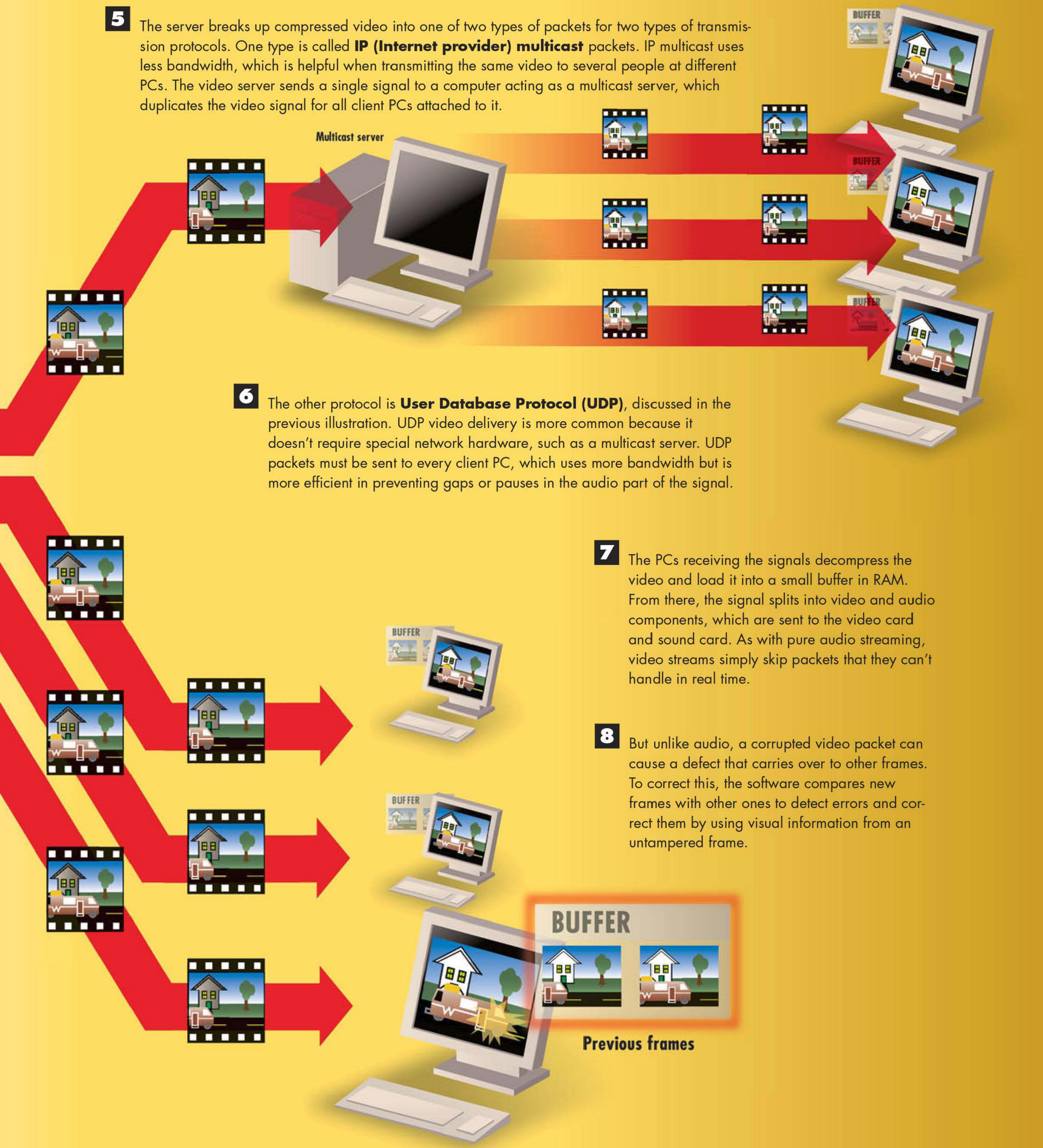
**2** The capture board sends the digital information through a **codec**, a **compression/ decompression** algorithm. Different codecs use several methods to compress the video.

**3** **Interframe compression** compares adjacent frames and transmits only those pixels that change from one to the other. When the camera is still, the background is not transmitted after a **key frame** has established what the background looks like. When the camera pans, causing the background to change, the entire frame is transmitted, creating another key frame. Thus, a still camera generally creates less data to transmit than a camera that is always moving.

**4** Codecs also skip frames for slower Internet links. The faster your connection, the more frames you receive and the smoother the video is.



**5** The server breaks up compressed video into one of two types of packets for two types of transmission protocols. One type is called **IP (Internet provider) multicast** packets. IP multicast uses less bandwidth, which is helpful when transmitting the same video to several people at different PCs. The video server sends a single signal to a computer acting as a multicast server, which duplicates the video signal for all client PCs attached to it.




**6** The other protocol is **User Database Protocol (UDP)**, discussed in the previous illustration. UDP video delivery is more common because it doesn't require special network hardware, such as a multicast server. UDP packets must be sent to every client PC, which uses more bandwidth but is more efficient in preventing gaps or pauses in the audio part of the signal.

**7** The PCs receiving the signals decompress the video and load it into a small buffer in RAM. From there, the signal splits into video and audio components, which are sent to the video card and sound card. As with pure audio streaming, video streams simply skip packets that they can't handle in real time.

**8** But unlike audio, a corrupted video packet can cause a defect that carries over to other frames. To correct this, the software compares new frames with other ones to detect errors and correct them by using visual information from an untampered frame.

# CHAPTER 30

# How the World Wide Web Works

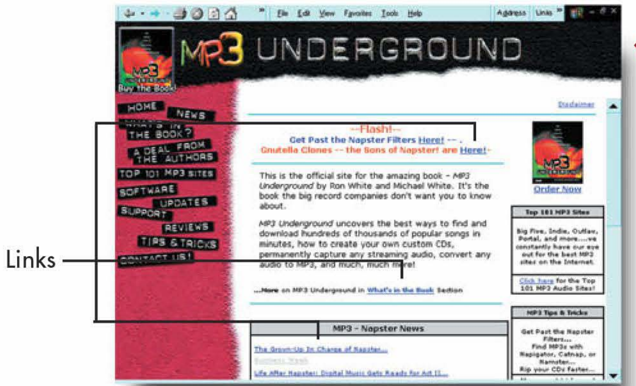
JUST a few editions ago when I got to this part of the book, I used to compare the World Wide Web to a Model A Ford. It got us where we wanted to go in cyberspace, but it was cantankerous and slow—slated for better things, to be sure, but some time in the far, far future. And no one today could imagine what enormous, pervasive effect the Internet is going…blah, blah, blah.

The far future's not as far off as it used to be. In just a few years, the Web has lost most symptoms of terminal crankiness. It's turned out to be a bomb for tasks that would have seemed a natural. (Ordered any groceries online lately?) And it's fulfilled other promises, but not in the way we expected. Previously, I wrote about the Web fulfilling the promise of Gutenberg, with everyone becoming a publisher. But I didn't expect them to be called blogs, nor that they would be as popular as they've become. Not long ago Web designers were cautioned not to use graphics, music, animation—anything that would make the Internet cool—because they choked pokey Internet connections. Now photos, Flash animations, and streaming video are commonplace. A decade or so ago, a couple of lawyers became nationally scorned by the pioneers of the Internet because they used the Web to advertise their practice—an obvious sacrilege. Today we don't make purchases larger than a pack of gum without consulting the Internet's consumer advice, sales, discount codes, and reviews by thousands of fellow consumers.

The Web is changing how we do everything and creating new standards for commerce, education, and communication. Want to know how much something is worth? Go to eBay. It's the ultimate free market, where the value of anything is quickly determined by a few bids that tell you—in real terms, not some economic gobbledygook—the most that anyone in the world would pay for a set of 2005 Chevy hubs.

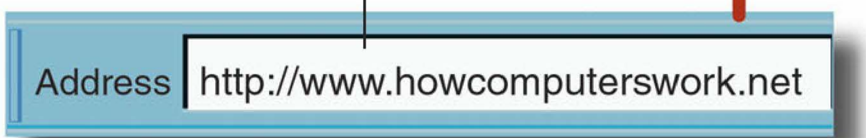
And then there is Google. Someone joked that Google is a god because it is omniscient and omnipresent. Religious issues aside, it has changed research, scholarship, and the settlement of bar bets around the world. Okay, it might not be a god, but certainly it's an oracle.

# How a Web Browser Opens Web Pages




**1** A website is a collection of files, documents, and graphics that someone has made generally available to others through the Internet. One way to begin a jump through the cyberspace of the World Wide Web is to click on a **hyperlink**. A hyperlink is a text phrase or graphic that conceals the address of a site on the Web. When the hyperlink is text, it is underlined and in a different color. The text doesn't have to be the actual address. These three words could hide a link to my page to supplement this book: www.howcomputerswork.net.

**2** Another way to direct the browser to a site is to type its **universal resource locator (URL)** into the address space on your browser's toolbar. For example, typing http://www.howcomputerswork.net aims your browser at my own Web site, where I have links to information on the topics in this book. Each part of the URL means something.




**http** identifies the site as one on the World Wide Web using HTML, or **hypertext markup language**. If the part before the slashes is **ftp**, that means the site is one that uses **file transfer protocol**; ftp exists primarily for plain text listings of files available for downloading, unadorned by the graphics and pizzazz of a Web page.

**://** alerts your browser that the next words will be the actual URL, which is broken up by periods. Each period usually is referred to as **dot**.

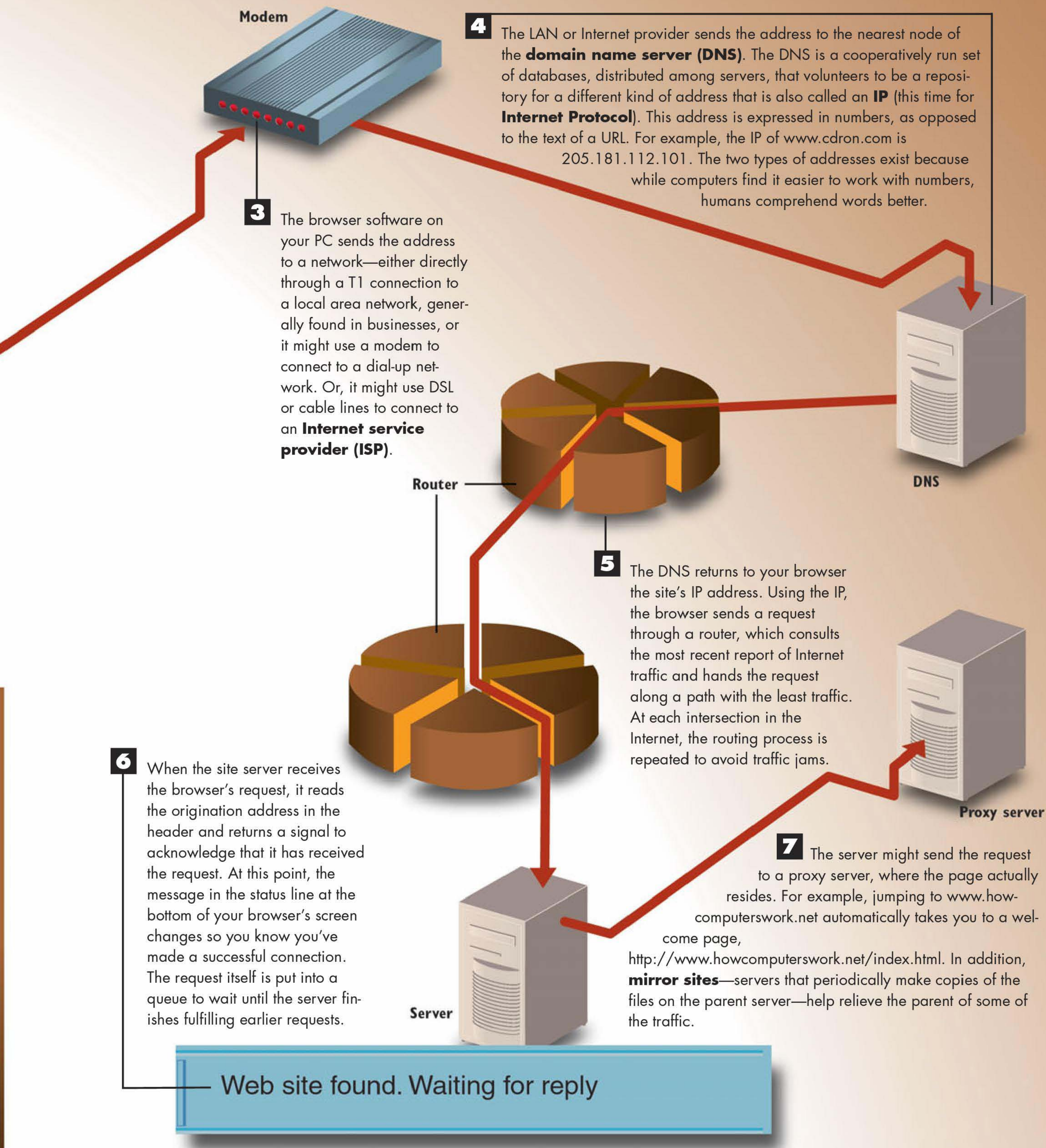
**www** identifies the site as part of the World Wide Web. The Web is a subset of the Internet that uses text, animation, graphics, sound, and video (and you don't really have to type www.)

**howcomputerswork** is the **domain name**. This is a unique name that must be registered with a company called Network Solutions, which has exclusive authority to register domain names under an agreement with the National Science Foundation. You must include a domain name.

**net** is the top-level domain name. In the United States, it indicates the purpose of the sponsors of the site. For example, net says that howcomputerswork is a network operation. And that I waited too long to get my first choice, ".com," which indicates a business. Other top-level names include **edu** for schools, **gov** for government offices, and the all-purpose **org** for organizations. Outside the United States, the top-level name might refer to a country, such as **uk** for United Kingdom.

**index.html** is a specific **page file** at the site, and the html tells the browser that the page uses the hypertext markup language—simple codes that determine the page's onscreen look.

**http://www.howcomputerswork.net/index.html**




**3** The browser software on your PC sends the address to a network—either directly through a T1 connection to a local area network, generally found in businesses, or it might use a modem to connect to a dial-up network. Or, it might use DSL or cable lines to connect to an **Internet service provider (ISP)**.

**4** The LAN or Internet provider sends the address to the nearest node of the **domain name server (DNS)**. The DNS is a cooperatively run set of databases, distributed among servers, that volunteers to be a repository for a different kind of address that is also called an **IP** (this time for **Internet Protocol**). This address is expressed in numbers, as opposed to the text of a URL. For example, the IP of www.cdron.com is 205.181.112.101. The two types of addresses exist because while computers find it easier to work with numbers, humans comprehend words better.

**5** The DNS returns to your browser the site's IP address. Using the IP, the browser sends a request through a router, which consults the most recent report of Internet traffic and hands the request along a path with the least traffic. At each intersection in the Internet, the routing process is repeated to avoid traffic jams.

**6** When the site server receives the browser's request, it reads the origination address in the header and returns a signal to acknowledge that it has received the request. At this point, the message in the status line at the bottom of your browser's screen changes so you know you've made a successful connection. The request itself is put into a queue to wait until the server finishes fulfilling earlier requests.

**7** The server might send the request to a proxy server, where the page actually resides. For example, jumping to www.howcomputerswork.net automatically takes you to a welcome page, http://www.howcomputerswork.net/index.html. In addition, **mirror sites**—servers that periodically make copies of the files on the parent server—help relieve the parent of some of the traffic.

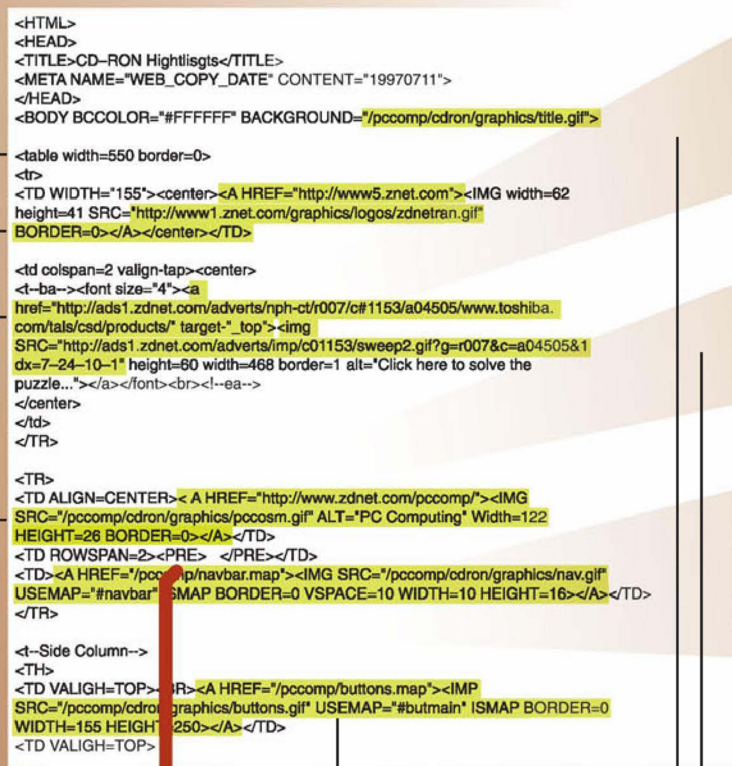
# How a Web Browser Displays Pages

**1** Stored on the server, the Web page itself consists of an HTML text file. HTML is a collection of codes enclosed in angle brackets–<>–that control the formatting of text in the file.

**2** The codes also can include the URLs of graphics, videos, and sound files that exist elsewhere on the server or on a different site entirely.

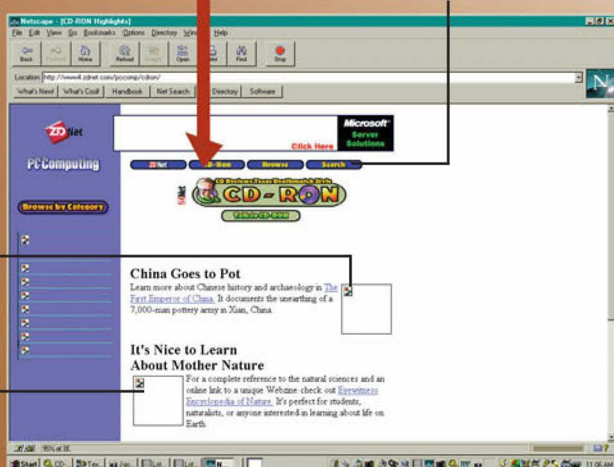
**3** When the server is free to respond to the browser request, it sends the HTML document back over the Internet to your browser's Internet provider address. The route it uses to get to your PC can differ vastly from the route your request followed to reach the server.

**4** At the same time, the server sends instructions to the sites that contain the graphics, sound, and video files identified in the page's HTML coding, telling those sites to send those files to your PC.

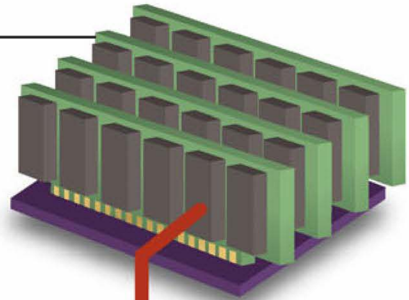


```
<HTML>
<HEAD>
<TITLE>CD–RON Hightlisgts</TITLE>
<META NAME="WEB_COPY_DATE" CONTENT="19970711">
</HEAD>
<BODY BCCOLOR="#FFFFFF" BACKGROUND="/pccomp/cdron/graphics/title.gif">

<table width=550 border=0>
<tr>
<TD WIDTH="155"><center><A HREF="http://www5.znet.com"><IMG width=62
height=41 SRC="http://www1.znet.com/graphics/logos/zdnetran.gif"
BORDER=0></A></center></TD>

<td colspan=2 valign-tap><center>
<t--ba--><font size="4"><a
href="http://ads1.zdnet.com/adverts/nph-ct/r007/c#1153/a04505/www.toshiba.
com/tals/csd/products/" target-"_top"><img
SRC="http://ads1.zdnet.com/adverts/imp/c01153/sweep2.gif?g=r007&c=a04505&1
dx=7–24–10–1" height=60 width=468 border=1 alt="Click here to solve the
puzzle..."></a></font><br><!--ea-->
</center>
</td>
</TR>

<TR>
<TD ALIGN=CENTER>< A HREF="http://www.zdnet.com/pccomp/"><IMG
SRC="/pccomp/cdron/graphics/pccosm.gif" ALT="PC Computing" Width=122
HEIGHT=26 BORDER=0></A></TD>
<TD ROWSPAN=2><PRE>   </PRE></TD>
<TD><A HREF="/pcc   p/navbar.map"><IMG SRC="/pccomp/cdron/graphics/nav.gif"
USEMAP="#navbar"   SMAP BORDER=0 VSPACE=10 WIDTH=10 HEIGHT=16></A></TD>
</TR>

<t--Side Column-->
<TH>
<TD VALIGH=TOP>   R><A HREF="/pccomp/buttons.map"><IMP
SRC="/pccomp/cdro   graphics/buttons.gif" USEMAP="#butmain" ISMAP BORDER=0
WIDTH=155 HEIGH   250></A></TD>
<TD VALIGH=TOP>
```

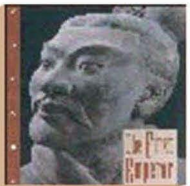
This code produces this graphic menu

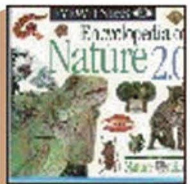



These are placeholders for graphics that have not yet been retrieved



**5** As the different parts of the page arrive at your PC, they are stored in a **cache**, a combination staging area and reservoir in the computer's RAM. Later, if your browser requests the same page or any of the elements on the page, such as a graphic, the browser retrieves it from the cache rather than going back over the Internet to the original sources.





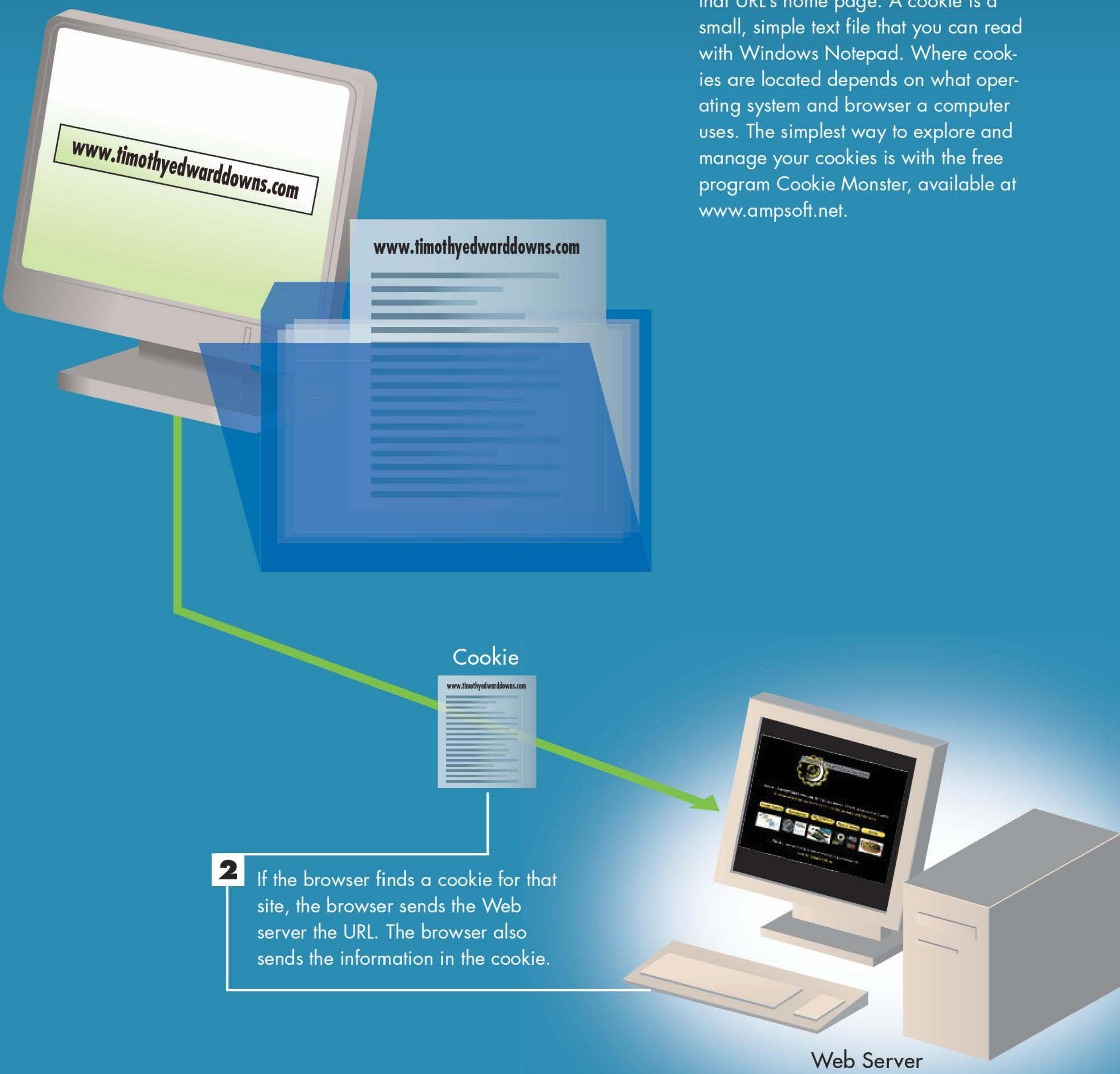

**6** If a non-streaming sound, music, or video file, such as a wave, MIDI, or AVI recording, is part of the page, the browser waits until all the file has arrived in the cache, and then it feeds it to Windows's Media Player, which uses your sound card to reproduce the sound.

**7** Meanwhile, the browser begins using the elements in the cache to reassemble the Web page onscreen, following the hidden HTML codes in the main document to determine where to place text, graphics, or videos onscreen. Because not all portions of the Web page arrive at your PC at the same time, different parts of it appear onscreen before others. Text, which is the simplest element to send, usually appears first, followed by still and animated graphics, sounds and music, and videos.

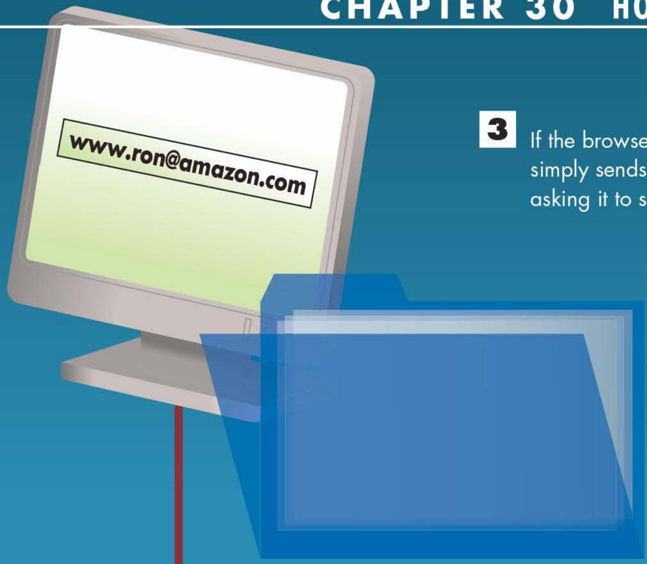

**8** Icons in the upper-right corners of both Netscape Navigator and Microsoft Internet Explorer are animated while parts of the page are still being received. When all elements have arrived and have been added onscreen, the animations freeze into still images, telling you that the browser has successfully retrieved the entire Web page.

# How Cookies Save Crumbs of Data

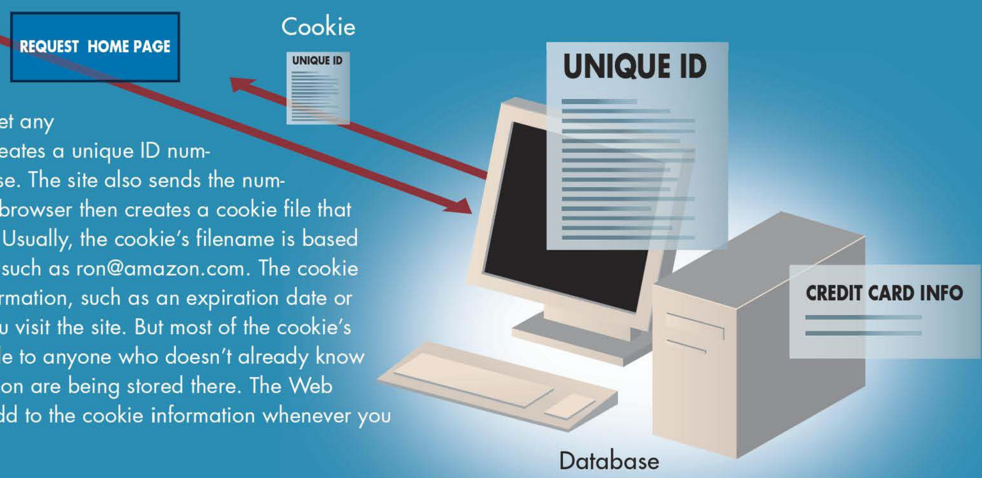
**1** When you type in a website's URL, your browser looks in a folder to see whether there is a cookie there associated with that URL's home page. A cookie is a small, simple text file that you can read with Windows Notepad. Where cookies are located depends on what operating system and browser a computer uses. The simplest way to explore and manage your cookies is with the free program Cookie Monster, available at www.ampsoft.net.




**2** If the browser finds a cookie for that site, the browser sends the Web server the URL. The browser also sends the information in the cookie.




**3** If the browser doesn't find the cookie, it simply sends a request to the Web site, asking it to send the site's home page.




**4** When the site doesn't get any cookie information, it creates a unique ID number for you in a database. The site also sends the number to the browser. The browser then creates a cookie file that contains the ID number. Usually, the cookie's filename is based on the name of the site, such as ron@amazon.com. The cookie might contain other information, such as an expiration date or a count of how often you visit the site. But most of the cookie's contents are unintelligible to anyone who doesn't already know what pieces of information are being stored there. The Web server can change or add to the cookie information whenever you request a page.

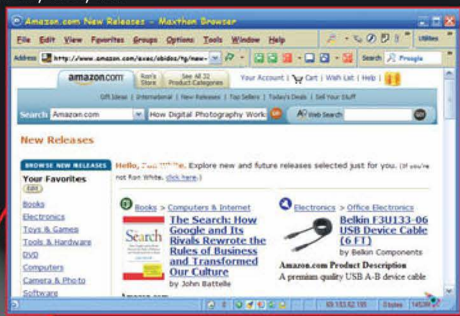
**5** Not all information is stored in the cookie. The site can keep information, such as your credit card number, but that type of information is encrypted and saved in the site's own database. Each time you visit the website, it recognizes you by the ID number in the cookie and uses the information in its database to automate such operations as filling in forms and one-click shopping.

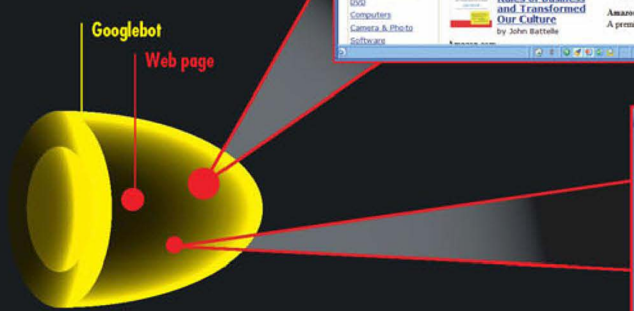

### What a Cookie Is Not

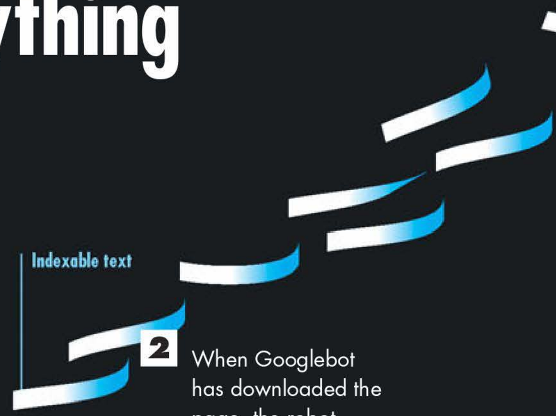
A cookie is only a text file—it is not a program. It cannot keep track of your computer operations or even what pages you visit. A cookie cannot ferret out personal information or documents on your computer. Basically, all a cookie can contain is information given to it by the Web server that placed it there. If you're still concerned about privacy, you can delete any cookies on your system, but be aware that you're giving up the convenience of automatic logins, easier online shopping, and pages customized to display your local weather and news.

# How Google Knows Everything

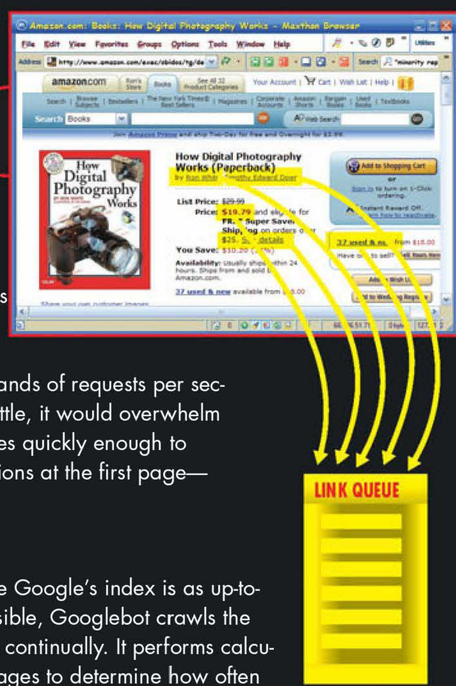
**1** Google is constantly **crawling** the Internet, indexing the information on the millions of pages found on the Web. Google uses another software program, called **Googlebot**, that, like a robot, works independently of its controller, Google itself. Googlebot asks a web server for a page, much as you might ask the server at Amazon.com to send you a page about a book you're thinking of buying.



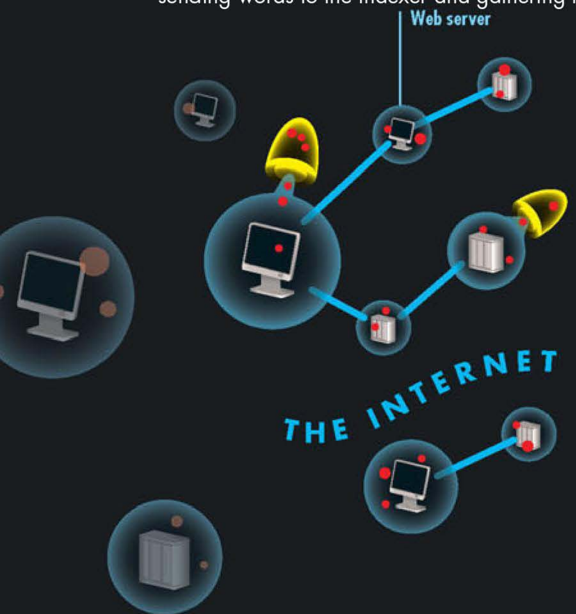



**2** When Googlebot has downloaded the page, the robot extracts the full text it finds there and sends it to Google's **indexing** program.




**3** The Googlebot then finds all the links on the page and adds them to a **queue**, a list of pages it (or its clones) will crawl. The Googlebot is not simply a single program visiting one page at a time. You get a better idea of how it works if you think of it as an army of Googlebots swarming throughout the Web. (Googlebot makes thousands of requests per second. It could crawl faster but it doesn't because if it operated full-throttle, it would overwhelm many web servers, and the servers would not be able to deliver pages quickly enough to users.) When a Googlebot crawls these new pages, it repeats its actions at the first page—sending words to the indexer and gathering links.




**4** To make sure Google's index is as up-to-date as possible, Googlebot crawls the same pages continually. It performs calculations on pages to determine how often they change and, based on the results, decides how often to crawl that site. Sites that frequently change, such as news sites, need to be crawled constantly throughout the day, and are called **fresh crawls**. The robot may crawl sites that rarely change just once a month.

CRAWL SCHEDULE

| PAGE | FREQUENCY |
|---|---|
| Amazon Home | 10 min. |
| CNN | 5 min |
| History of Aardvarks | Monthly |
| New York Time Page 1 | Daily |


**5** The **indexer** receives the text and stores it in its database. The index is sorted alphabetically by search term. Each index entry contains the list of pages on which the term appears. The indexer doesn't index commonly used words, called *stop words*, such as *the*, *on*, *is*, *or*, *of*, and *why*, and doesn't store single digits, single letters, and some punctuation marks.

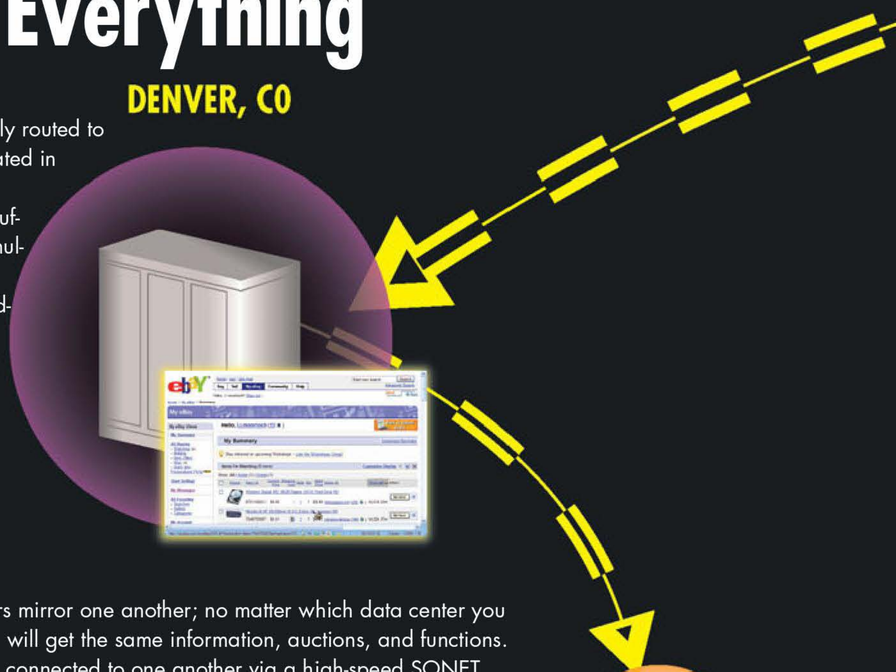
**6** When you type in your query, it's sent to Google's web server. The web server forwards it to Google's index servers.


**7** Google's index servers match your search term to the most relevant documents that contain the term. The method Google uses to match queries to documents is Google's secret sauce, the key to its ability to return the most relevant results.


**8** Google uses hundreds of factors to decide which documents are most relevant, but the best known is Google **PageRank**, a rating based on how often others visit a page. Google also considers where it found the search terms on the page and, if you use multiple search terms, how close those terms are to one another. If popular pages link to a page, that page will have a higher rank than if it is linked to from only unpopular pages.

**9** When the index server determines the results of the search, it sends a query to Google's **doc servers**. They retrieve the stored documents, which include site names, links, and snippets that summarize each page. The doc servers send the results to the web server, which sends the results to you.

# How eBay Sells Everything



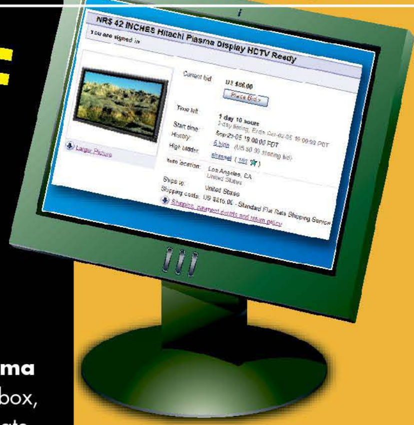

**1** When you go to the eBay site, you are automatically routed to one of four eBay data centers. Two centers are located in Santa Clara, California; one is in Sacramento, California; and one is in Denver, Colorado. eBay suffered a series of disastrous outages in 1999. The multiple centers ensure eBay will always be up and running to serve 95 million buyers and sellers worldwide.


**2** The data centers mirror one another; no matter which data center you connect to, you will get the same information, auctions, and functions. The centers are connected to one another via a high-speed SONET (Synchronous Optical Network) fiber-optic network.

**5** The search servers send the search request to a cluster of 50 database servers running an Oracle database on top of Sun SPARC hardware. This database is in essence what eBay really is—it contains all the details of every single auction on eBay. With more than 5 million items for sale every day, the database requires organization in the form of 27,000 unique categories. Eight of those categories sell more than a billion items annually, including toys, clothing, accessories, collectibles, sports, books, movies, and music. Consumer electronics is a $2 billion category, and computers sell $2.1 billion. The largest category in sales is eBay Motors, which racks up $5.3 billion a year.

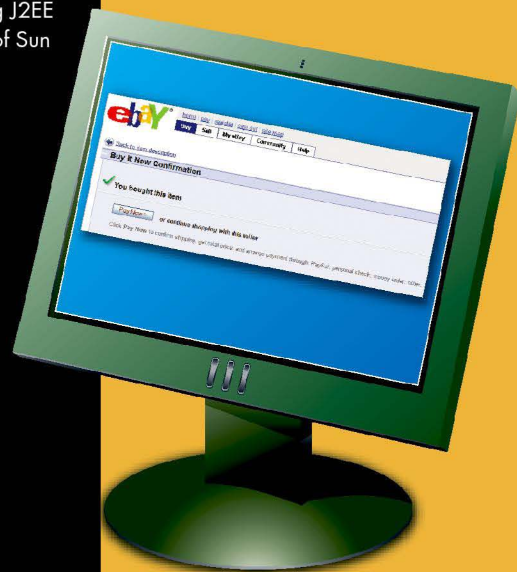
**8** When the transactions are complete, the buyer and seller are tasked with the all-important duty of rating the transaction experience. These ratings are crucial to eBay because they help to assure its users that they are not dealing with swindlers or deadbeats.

## HDTV PLASMA DISPLAYS




**3** When a buyer logs on to eBay to find a bargain, the system assigns him to one of the web servers at the data center. The servers run Windows XP Server and Windows 2000 server software.

**4** If the buyer types **HDTV plasma displays** in the eBay search box, the request is shunted to separate search servers. These computers run an application written using J2EE (Java) and running on top of Sun Microsystems hardware.




**6** The search results show all the plasma displays on auction. If the buyer bids on one of them, eBay's auction software compares his bid to any others. A bid for more than the minimum needed to top the previous winning bid might be partially hidden, with only the amount needed to win being shown. But if that bid is topped later, the hidden portion may be used to regain the bidder's lead.


**7** The time a bid is placed is also recorded in the database in case of ties. Flags in the database signal when the auction ends. The database sends all the information about the winning bid to the application server. The application server sends the information to the web server so it can be posted for all to see, and it also automatically creates emails to the winning and losing bidders and the seller. The sale costs the seller about $2 plus 1.5% of the sale price.

CHAPTER 31

# How Internet Security Fights Off PC Invaders

WE should be grateful that most computer crackers are grossly egotistical. The very premise of breaking into a megacorporation's giant computer, leaving a virus, or substituting a screen of his own scatological design for the company's home page—that's done by someone who wants attention. They can't resist leaving a program or note that's a digital sign of Zorro, something to let you know that while you might own the computer, the cracker owns its soul.

I'm using the term **cracker** here instead of **hacker**, which most people think of when they hear about some teenager who broke into the high school's computer and gave himself a lot of A's. Self-described hackers break into computer systems out of curiosity or to test their skills, without stealing information and without trashing the hacked system. Hackers look down on crackers as immature, unsophisticated kiddies.

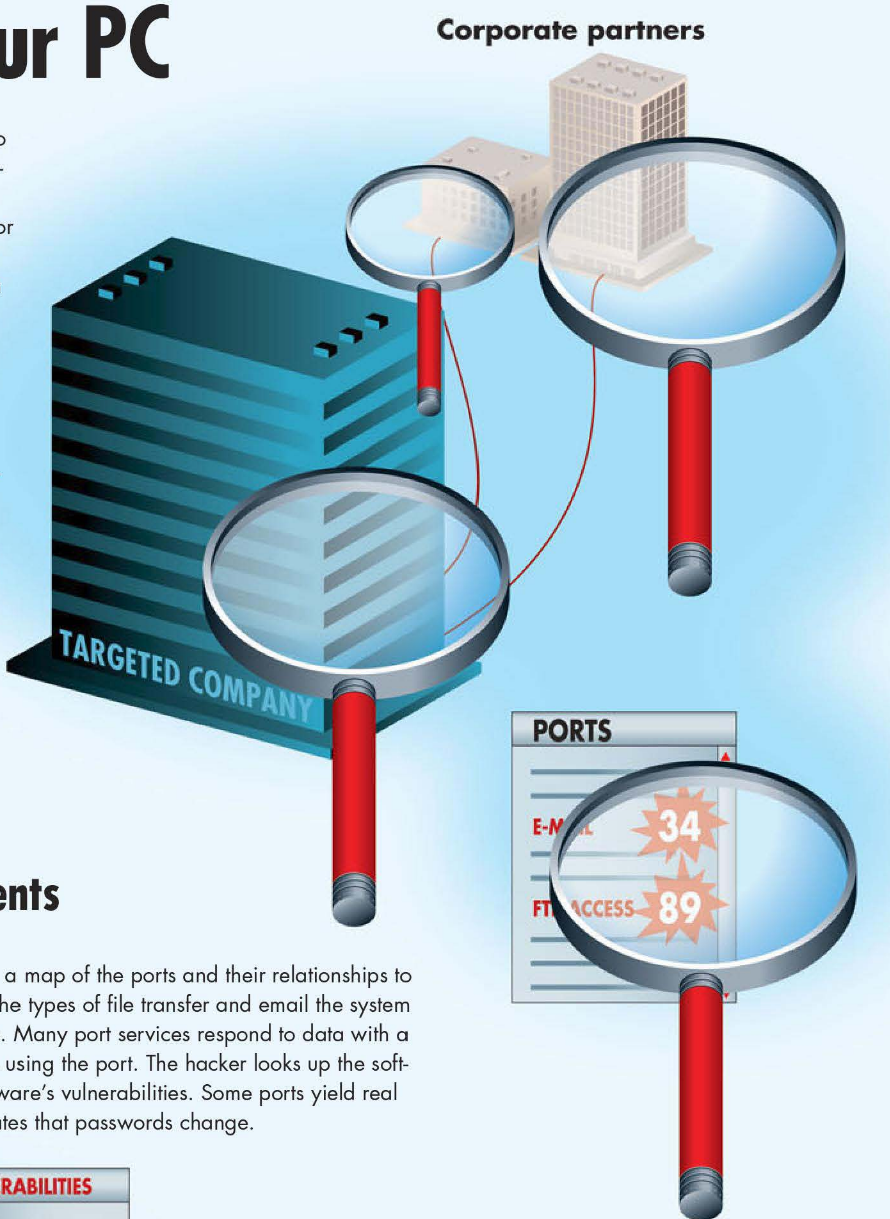
Crackers and hackers have one thing in common: Both must have endless patience, a high tolerance for delivery pizza, and a dedication to their life's calling that you rarely find outside of a monastery. It means starting as a *script kiddie*, running program scripts written by others that sniff and nudge and try to bluff their way past a computer's defenses. The world of hacking is ultra cool and ultra exclusive. You will only be admitted—make that tolerated—when you show that you've mastered those scripts and even written some of your own. You might take on a monster crack to prove you're up to snuff.

This proving yourself is the undoing of many crackers. It's difficult to do a crack and leave no clues behind—if only because the up-and-coming cracker just has to let someone know he did it. The only payoff in this game is recognition. The only way you earn it is to have encyclopedic knowledge of network security, including all the holes and back entrances that not even the people running a network know about.

Fortunately, while no system, from a home PC to an Internet-connected corporate supercomputer, is truly crack-proof, as a user there are tools at your disposal that help you keep control of your PC. Firewalls and antivirus software are readily available to help you keep control of your PC where it belongs—in your hands.

Still, the next time you tune in to the news and learn about a new virus, a denial-of-service attack that overwhelms an organization's incoming lines, or a crash that spreads like a California wildfire, remember this: These are the cyber-attacks you don't have to worry about. A really dangerous cracker would keep his activities secret, and limit himself to, say, small transfers of money, none so large as to attract attention. When the real master cracker comes along, no one will know anything's going on. In fact, it could be going on right now, but you'll never really know if he's stopped.

# How Computer Hackers Break Into Your PC




**1** Unlike in movies, where hackers break into a computer in minutes with only a few keystrokes, hacking deep enough into a computer to take control of it might take days or weeks. The computer hacker methodically follows a set of procedures that pries open a crack wider and wider with each step.

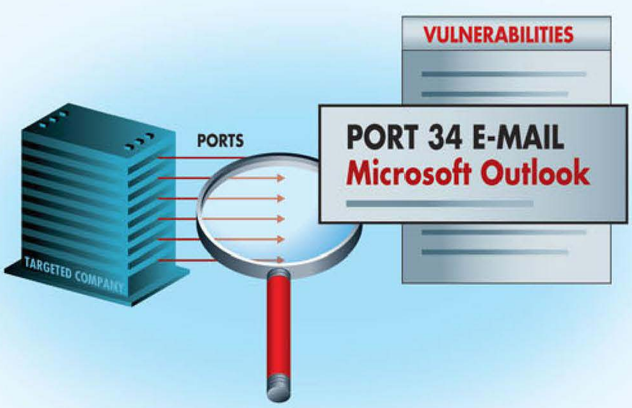
## Casing the Target

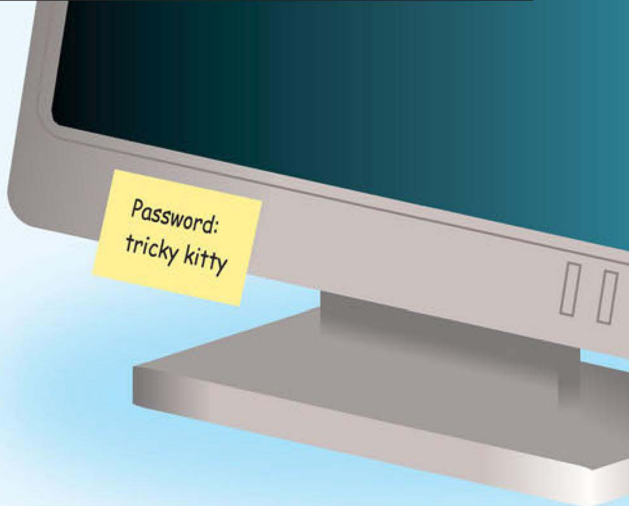
**2** The hacker performs a **footprint analysis** of the intended target using publicly available information, such as its size, subsidiaries, and vendors that might have access to the target's computers.

**3** Using readily available hacking software, he scans the target computer's **ports** for potential break-in points. Ports are numbers used to identify different services the computer provides, such as network input/output and email.

## Identifying Target Components

**4** Based on the analysis, the hacker creates a map of the ports and their relationships to each other. He uses this to try to identify the types of file transfer and email the system uses by sending random data to the ports. Many port services respond to data with a **banner** that identifies the software that's using the port. The hacker looks up the software in online databases that list the software's vulnerabilities. Some ports yield real pay dirt in the form of user names and dates that passwords change.






## Obtain Access

**5** To gain access to the target system, the hacker has two approaches. The low-tech method involves contacting employees to trick them into revealing their passwords. The hacker might call pretending to be part of the IT help staff, claiming that a security malfunction requires the employee to verify her password. A daring hacker might visit the offices in person to look for passwords employees have taped to their monitors.


**6** The second method is a **brute force** attack. The hacker uses a hacking program to try to log onto the system with the usernames she's acquired. When the system asks for a password, the program responds with a word from a list of likely passwords, such as *opensesame* or sexy. The program repeats the process until the list is exhausted, it chances upon the right password, or the host locks the user out for too many failed attempts.

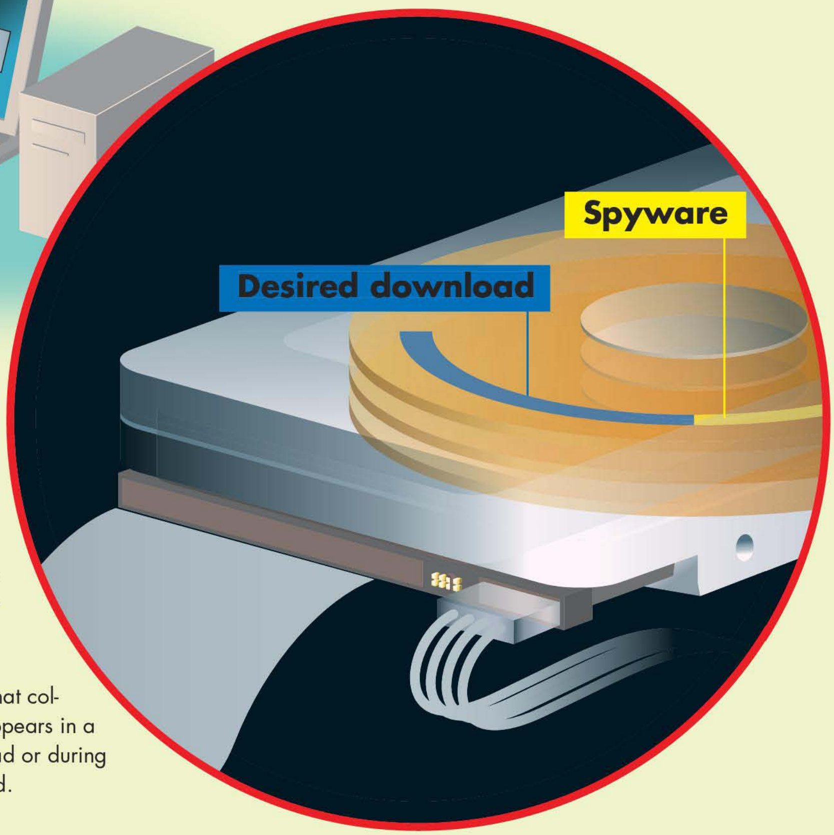

## Escalate Privileges

**7** After the hacker has entered the system with user-level privileges, he looks for passwords of higher-level users that grant greater access to the system. Good sources are registry keys and email.

**8** Finally, with access to the most secret ranges of the network, the hacker uploads innocuous-seeming **trojan** programs to one or more of the computers on the network. These programs appear to the human eye or a virus scanner to be ordinary, harmless files. In actuality, they are programs that open a **back-door** through which the hacker may now enter the network at will.

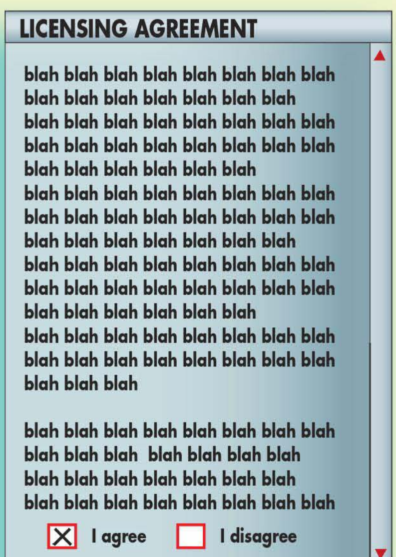
# How Spyware Reports Everything You Do


**1** **Spyware** is a program on your hard drive that sends information about you and how you use a PC to marketers and advertisers, usually without you being aware it's happening. **Adware** is a variant of spyware that displays advertising for products it thinks might interest you based on your website browsing habits.




**2** A web link might launch a spy program to your hard drive when you click the link. Normally, however, spyware enters your system when you download free software such as KaZaA and Alexa. Installation of the program you wanted also installs the spyware. Antivirus software and firewalls do nothing to block spyware installed with a download you ask for.

**3** A warning that you have installed a program that collects information from your computer usually appears in a **Privacy Policy** where you found the download or during installation. Such fine print is notoriously unread.



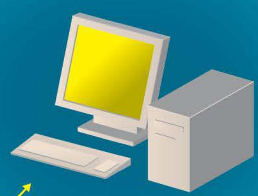


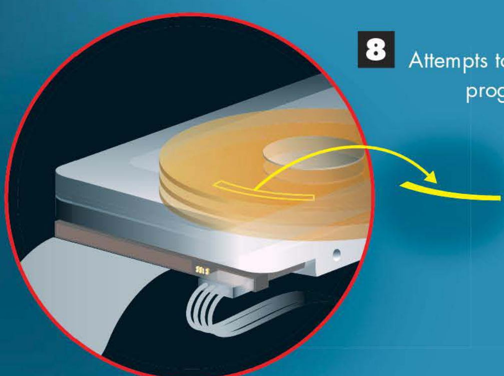
**4** Most spyware is installed as just another program in the Program Files folder. Other spyware embeds itself in the Windows Registry, where it is harder to find and more dangerous to extract.

**5** The spyware program begins gathering information. Some programs, such as Comet Cursor, limit themselves to generic information and don't collect any data that would identify you. But other types of spyware might record all your keystrokes (including any password you type), snoop through documents you create with a word processor or spreadsheet, track each page you visit on the Web, or scour your files for anything that looks like a credit card number. The information might be stored in a **cookie**, but cookies in themselves are not harmful and don't indicate the presence of spies.


**6** Spy programs aren't limited to harvesting data. Spyware conceivably does anything your computer can do. It can insert websites in your browser's bookmarks and change your home page.

**7** The spyware opens a **backchannel**—an Internet link hidden in the background of your computer's operations and leads to whoever created the spy. It uses the backchannel to send information found on your computer. If the program is adware, it uses the information it obtains to display advertising that matches your interests.

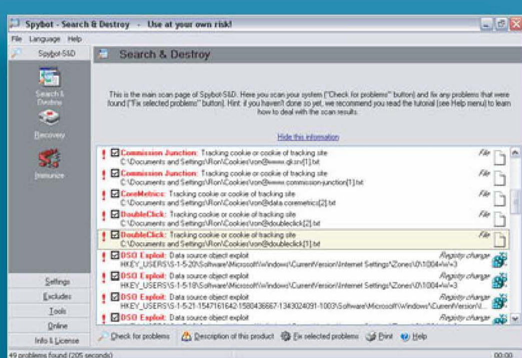




**8** Attempts to remove spyware are often thwarted. Some spy programs include **tricklers**, which re-install the spy files as fast as you delete them. At the least, uninstalling usually causes the program you downloaded with the spy to stop working.

### Counter Spies

Don't try to remove spyware on your own—it'll outsmart you every time. If your system doesn't have its own spyware management tool (Vista includes its own, for example), get a program such as Spybot Search and Destroy, available through http://www.howcomputerswork.net. Spybot is the best of the spy assassins, but even it can't uncover all spyware.

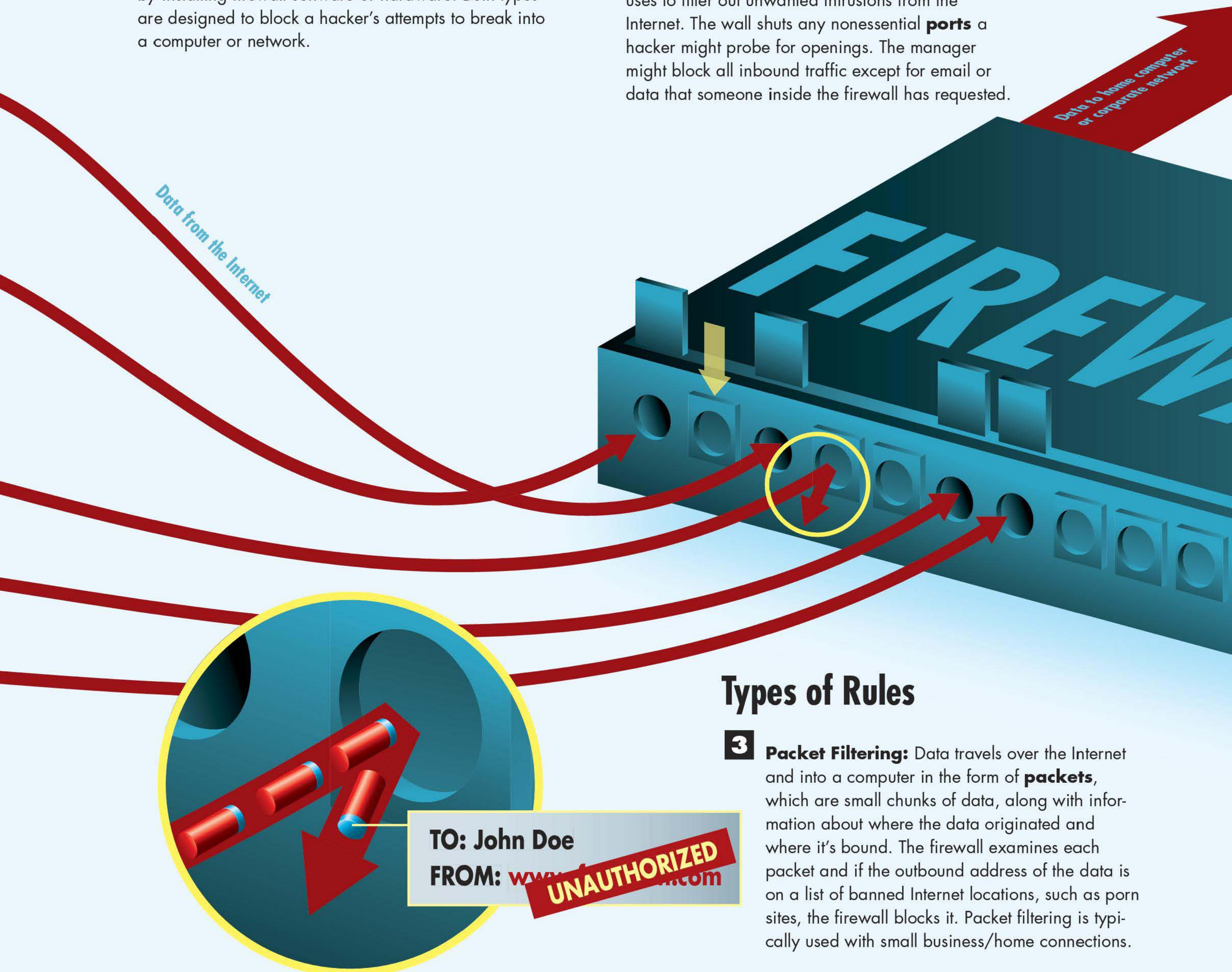


# How Internet Firewalls Keep Out Hackers

## Setup

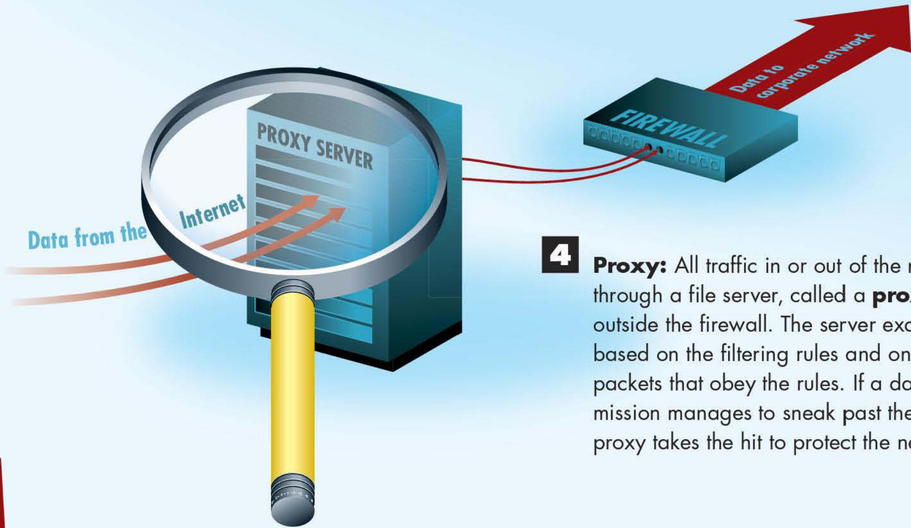
**1** Computer owners or network managers place a **firewall** between their computers and the Internet, either by installing firewall software or hardware. Both types are designed to block a hacker's attempts to break into a computer or network.

**2** The firewall's manager sets up **rules** the firewall uses to filter out unwanted intrusions from the Internet. The wall shuts any nonessential **ports** a hacker might probe for openings. The manager might block all inbound traffic except for email or data that someone inside the firewall has requested.

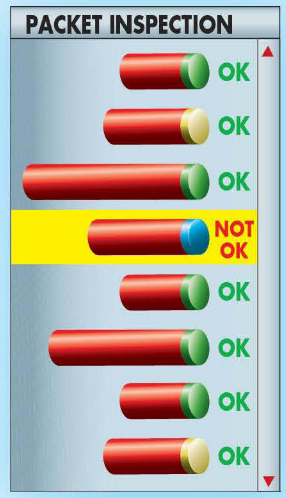



## Types of Rules

**3** **Packet Filtering:** Data travels over the Internet and into a computer in the form of **packets**, which are small chunks of data, along with information about where the data originated and where it's bound. The firewall examines each packet and if the outbound address of the data is on a list of banned Internet locations, such as porn sites, the firewall blocks it. Packet filtering is typically used with small business/home connections.



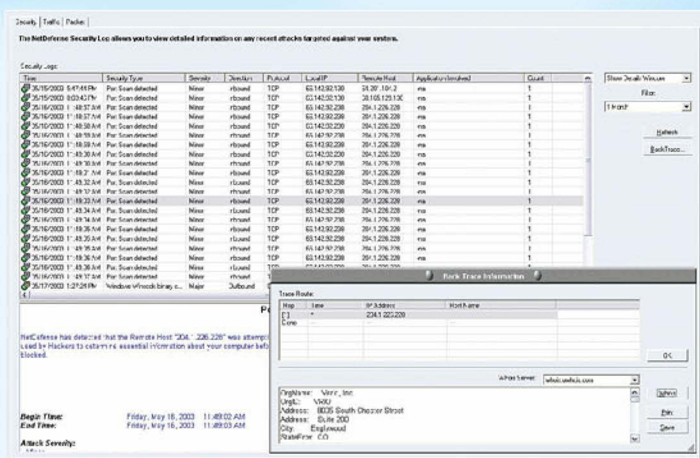

**4** **Proxy:** All traffic in or out of the network goes through a file server, called a **proxy**, located outside the firewall. The server examines all data based on the filtering rules and only forwards packets that obey the rules. If a dangerous transmission manages to sneak past the filters, the proxy takes the hit to protect the network.




**5** **Stateful Inspection:** The firewall compares key parts of each packet to a database of known safe data. The packet data must resemble data the firewall has seen before. The wall sends incoming data that passes muster to its final destination. Packets that fail the test are discarded by letting subsequent data packets write over them.

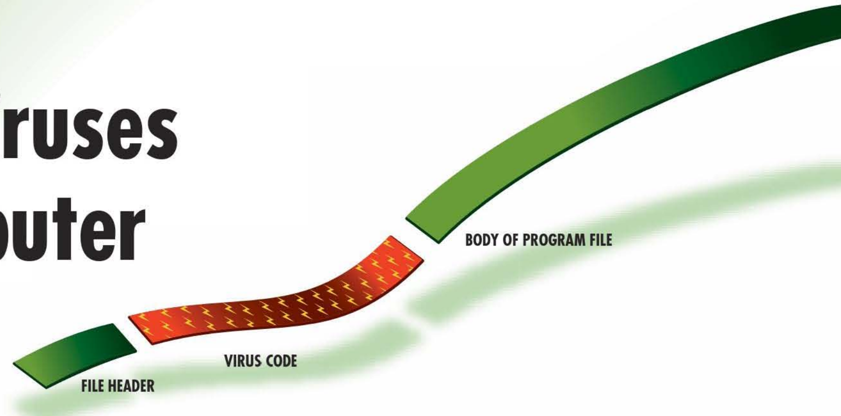
## Alerts


**6** When a firewall detects suspicious activity, it sends an **alert** in the form of a pop-up window or email to notify the computer's user or the network manager that someone might have tried to break in.



**7** The firewall also adds the intrusion to a **security log**, including information about the type of attack and the IP address of the computer sending the intrusive code. Typically, the firewall also saves records of packets that have gone in and out of a computer. This information can be reported to a user's Internet service provider (ISP) or to the computer help staff.

# How Computer Viruses Invade Your Computer



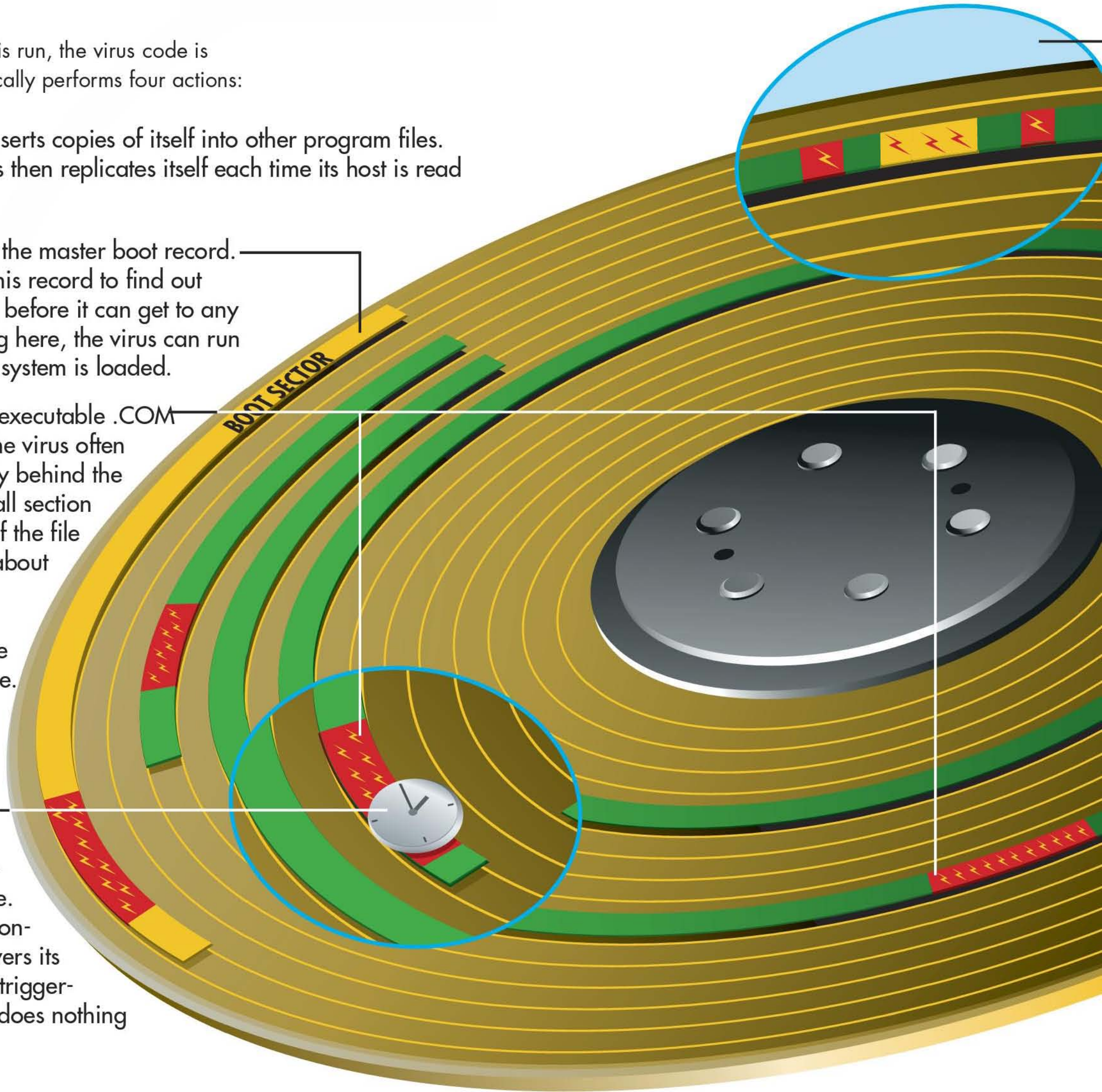

**1** A **virus** is created when a programmer intentionally infects a program or disk with computer code that has the capability to replicate itself, hide, watch for a specific event to occur, and deliver a destructive or prankish payload.

**2** When the infected program is run, the virus code is executed first. The code typically performs four actions:

**Replication:** The virus inserts copies of itself into other program files. Each descendant of a virus then replicates itself each time its host is read by the computer.

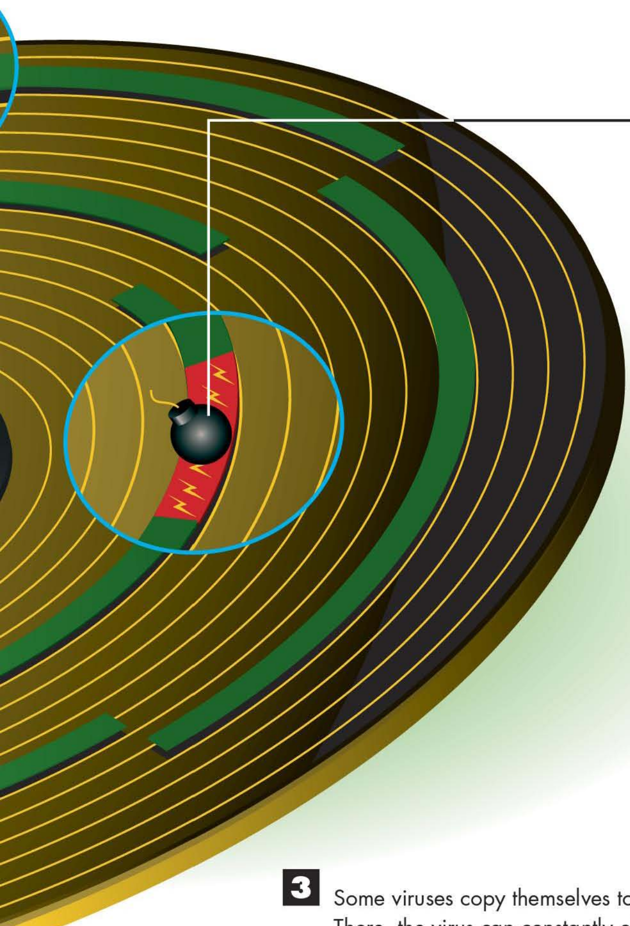
*Boot record viruses* target the master boot record. The computer must read this record to find out how the disk is organized before it can get to any of the other files. By hiding here, the virus can run even before an operating system is loaded.

*PProgram viruses* look for executable .COM and .EXE program files. The virus often inserts its copy immediately behind the program's **header**, a small section of code at the beginning of the file that contains information about what kind of file it is. This ensures that the virus is always executed before the legitimate portion of the file.

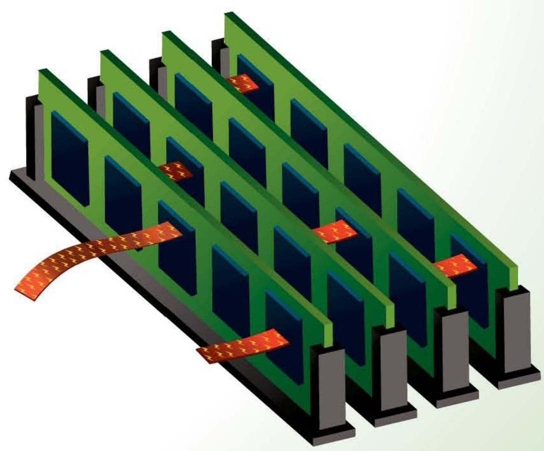



**Event watching:** Everytime the virus runs, it checks for a certain condition, usually a specific date. Whenever the triggering condition exists, the virus delivers its destructive payload. If the triggering event is not present, it does nothing but replicate itself.

**Camouflage: Stealth viruses** disguise themselves to avoid detection by antivirus software. The disguises used by a morphing virus consist of nonfunctioning, changing sections of fake code interspersed among working sections of the virus. Each time the virus replicates, it creates different fake code to break up its identifying signature. (See the facing page for more information on signature detection.) The virus might also falsify information in the header about the file's length so the program file appears to be the correct size.

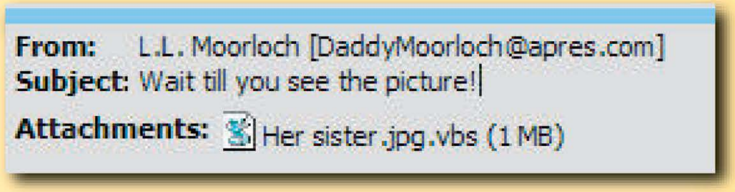


**Delivery:** When the triggering condition is met, the virus unleashes its payload—the operation that is its *raison d'etre*. The payload might be harmless, such as displaying a "you've been had" message. Or, the payload can be destructive, erasing or scrambling files or information on the drive that tells the operating system how to find files on the disk. The most insidious viruses are those that do not announce their presence and make subtle changes to files. It could, for example, randomly change numbers in an accounting program, steal passwords, or introduce delays to make a computer run slower.

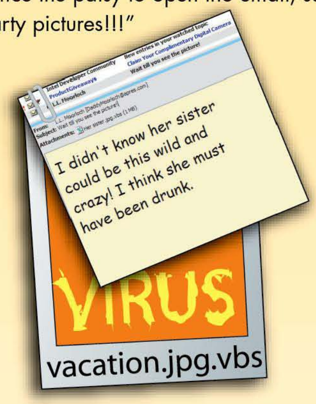


**3** Some viruses copy themselves to memory. There, the virus can constantly check for a triggering action such as certain keystrokes. The **memory-resident virus** can also watch for attempts by antivirus software to find infected files and return phony information that hides the virus from detection.

# How Viruses Travel in Your Email

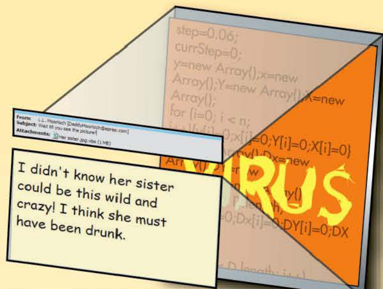



**1** An unsuspecting victim receives an email that appears to have been sent by someone the victim knows, and the subject of the message is worded to entice the patsy to open the email, such as "Wild party pictures!!!"

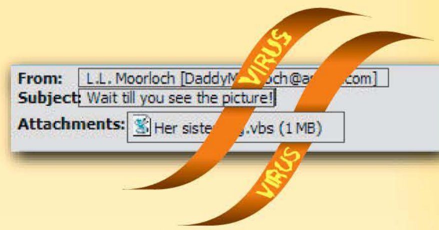



An **attachment virus** is a program attached to an email message. It pretends to be a photo or movie the victim can view on his computer. The name of the attachment is disguised to hide its true nature. For example, the attachment could be vacation.jpg.vbs. Many users notice the "jpg" and assume it's a vacation photo without realizing the "vbs" identifies the attachment as a Visual Basic script, a type of program. Attachments are the most common type of virus. Examples: Melissa, LoveLetter, and AnnaKournikova.

**2** Hidden within the email is one of three types of viruses.



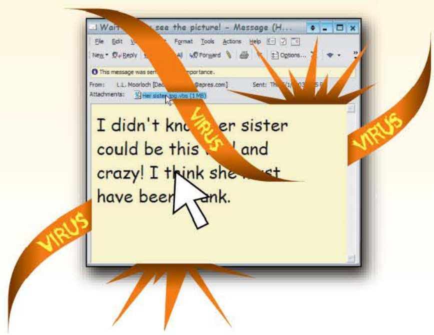

An **HTML virus** is **active content** code, essentially a small program written in **JavaScript** or **ActiveX** software languages. Active content is used on the Web whenever you buy something, fill in forms, vote in a poll, or take part in any other interactive pages on the Web. The HTML virus is not displayed when you open a message formatted in HTML. Examples: Kak worm, BubbleBoy, and HapTime.



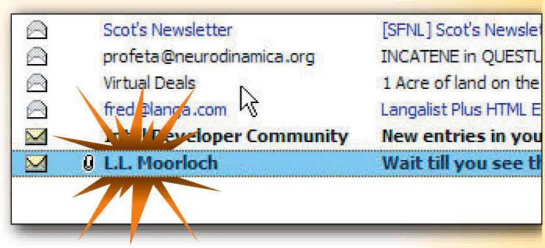

A **MIME (Multi-Purpose Internet Mail Extensions) virus** takes advantage of a security hole in Outlook Express and Internet Explorer. The perpetrator fills in forms in the email's header with more information than the header can hold in its **buffer**—memory reserved for the form entries. When the buffer runs out of room to hold the entry, the **overflow**—the virus—spills into **stack** memory being used by the microprocessor to run programs, and the virus is executed instead of legitimate code. Example: Nimba.

**3** What makes the virus launch its attack depends on what type of virus it is.


The **attachment virus** runs only when the victim double-clicks the attachment's filename.

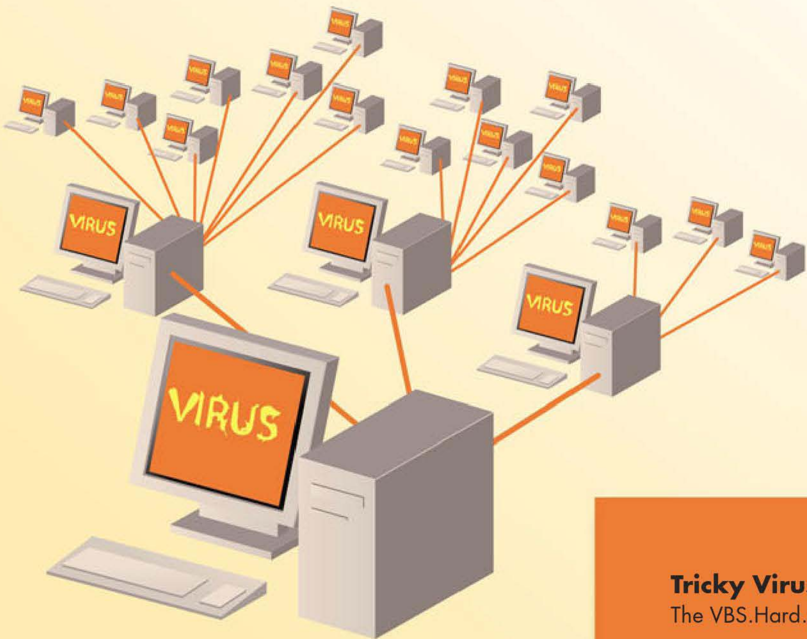



The **HTML virus** jumps into action when the victim opens the message to read it. Merely displaying the message in the preview window also launches the virus.




The **MIME virus** can run even if it's not seen. Part of the code hidden in the header tells Outlook Express that the message is a .wav file—a Windows audio file. Outlook Express automatically executes the virus without the victim doing anything.


**4** Viruses hidden in email do different kinds of mischief, but the first thing any of them does is propagate itself. It searches the victim's address book, old email, even documents created with Word or Excel. From these, it extracts names and email addresses.




**5** The virus uses the addresses to send duplicates of itself to the victim's friends and business acquaintances, hidden in the same email it rode into the computer. To make itself harder to trace, the virus might pick a name at random from the address book and put that in the From: field. In minutes, the virus spreads itself to hundreds of other computers, sometimes accompanied by haphazard attachments of letters and spreadsheets the virus has found among the victim's files.

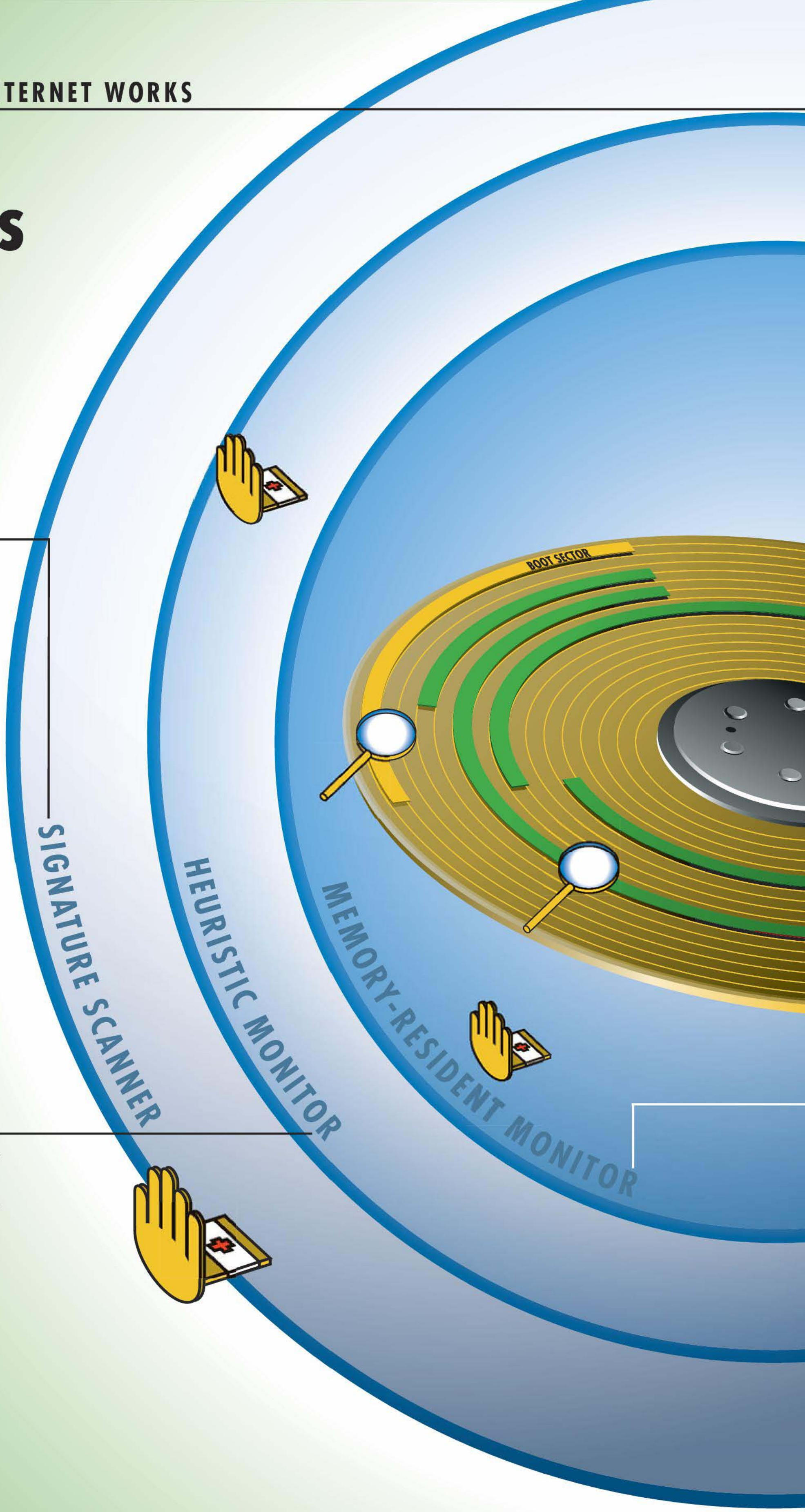
**6** Eventually, thousands of copies of the original virus deliver their payloads, which are anything from taunting messages to erasing hard drives. They might dump their payloads right after they've finished replicating themselves, after so much time has passed, or they might all go off at the same time on the same day.

### Tricky Viruses

The VBS.Hard.A@mm virus is an attachment that comes on an email message, warning against a nonexistent worm called VBS.AmericanHistoryX_II@mm. The subject of the message is "FW: Symantec Anti-Virus Warning;" the message promises more information in the attached memo.

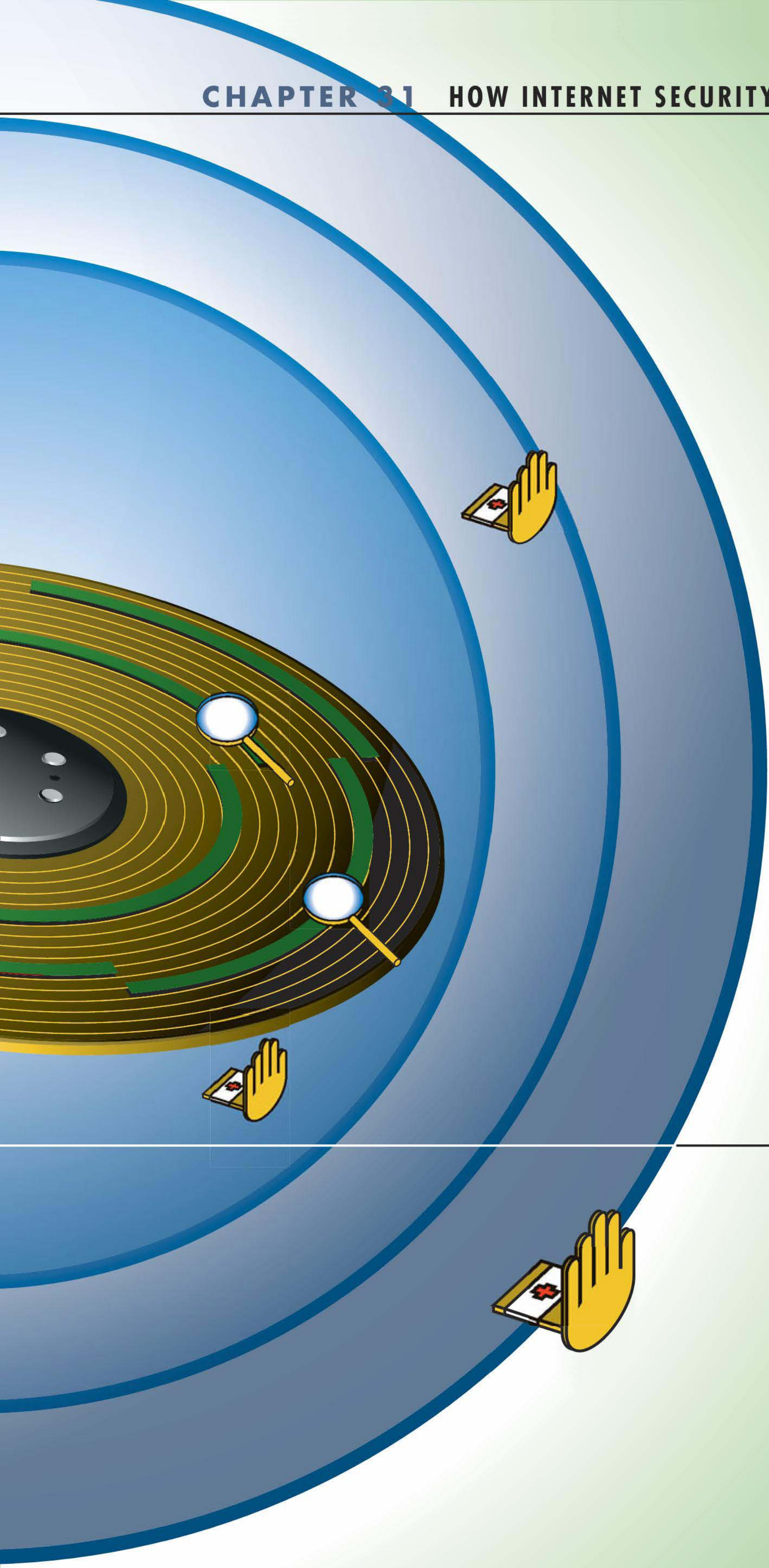
When the attached file, www.symantec.com.vbs, is opened, it makes the Internet Explorer home page a phony website that warns about a fictitious worm. It also makes Outlook send copies of the bogus virus warning to everyone in the address book. Each November 24, infected computers display the same message: "Don't look surprised! It is only a warning about your stupidity. Take care!"

# How Antivirus Software Fights Back




**1** The first line of defense against viruses is **antivirus software** that inspects the master boot record, program files, and macro code for the presence of viruses. **Signature scanners** look at the contents of the boot record, programs, and macros for telltale sections of code that match viruses contained in a table of all currently known viruses. Such tables must be updated regularly to be effective against new viruses.

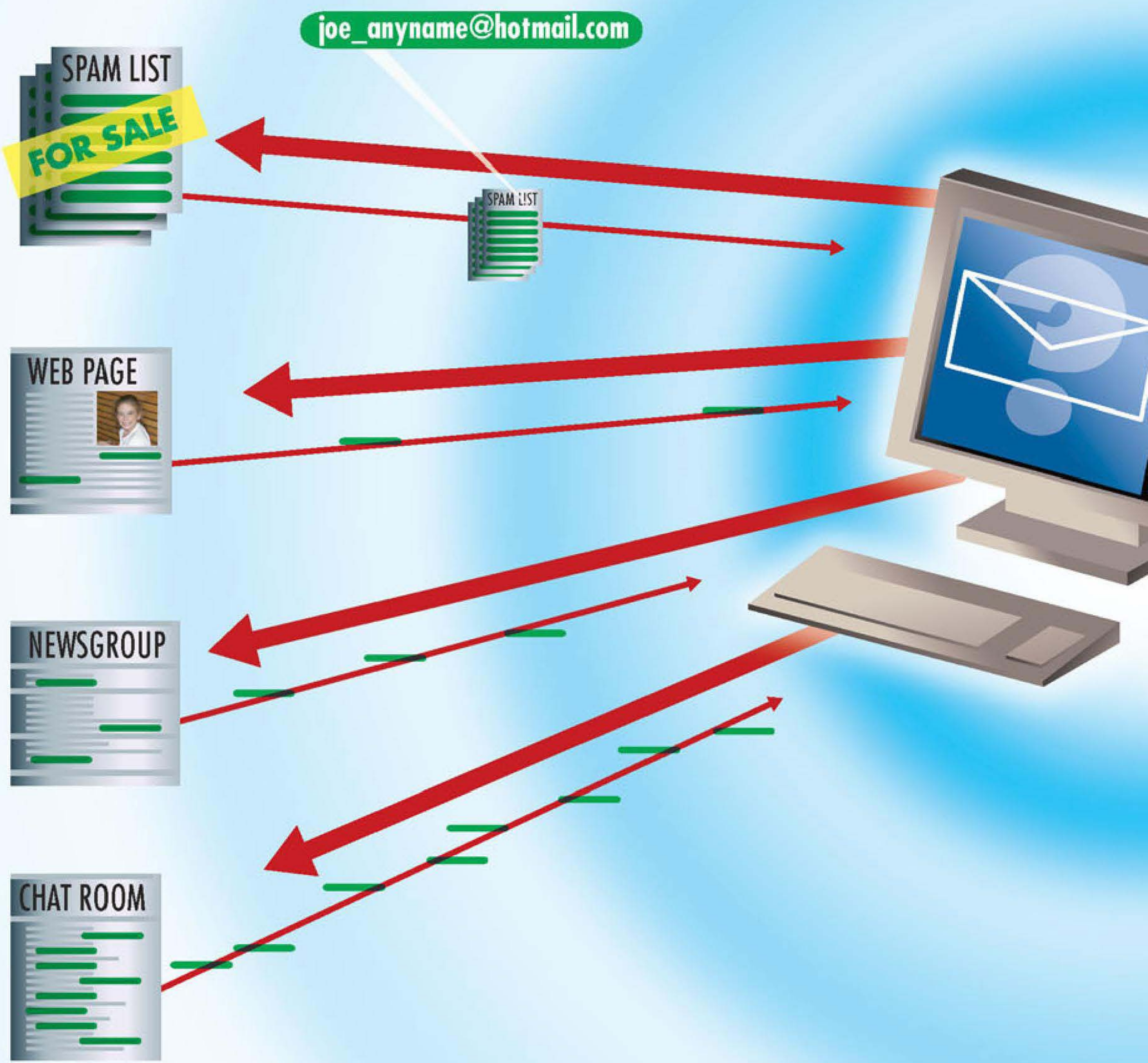
**2** Because stealth viruses evade detection by signature scanners, **heuristic detectors** look for sections of code triggered by time or date events, routines to search for .COM and .EXE files, and disk writes that bypass the operating system.



**3** **Memory-resident** antivirus software installs programs in RAM that continue to operate in the background while other software applications are running. These programs monitor all the computer's operations for any action associated with viruses, such as downloading files, running programs directly from an Internet site, copying or unzipping files, attempting to modify program code, or programs that try to remain in memory after they're executed. When they detect suspicious operations, the memory-resident programs call a halt to operations, display a warning message, and wait for the user's OK before allowing the task to continue.

# How Spammers Find You

## Public Sources for Spammers




1. Not all spammers compile their own spam lists. Many buy lists from others, who use a variety of techniques to harvest email addresses or potential email addresses, and then sell those lists to the highest bidder.

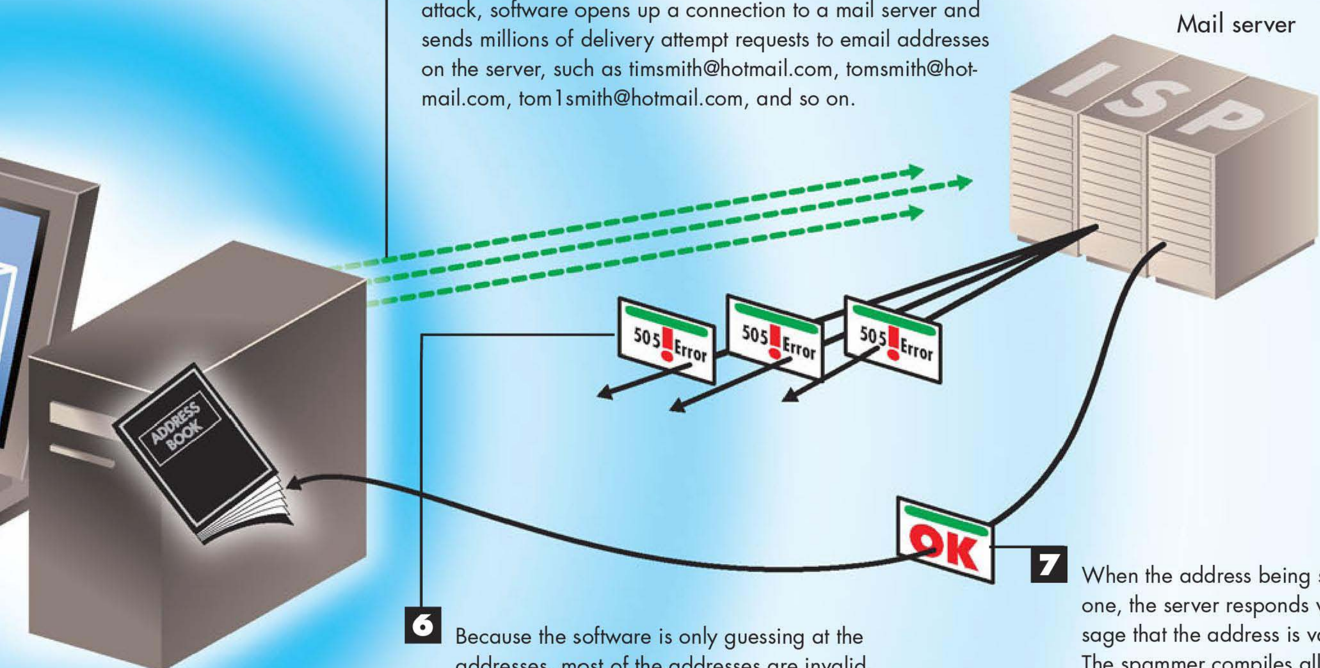
2. Spammers send out automated spiders across the Internet that crawl across web pages looking for email addresses in mail:to links or posted on the page. The spiders send all the addresses back to the person compiling the spam list.

3. Spiders also look through Usenet newsgroups for email address, and when they find them, they send the addresses back to the spam list compiler.

4. Other kinds of spiders visit chat rooms and grab all the email addresses of those in the rooms.

## Harvesting from Email Directories

**5** A dictionary attack, also called a directory harvest attack, is a common technique for harvesting email addresses. It is used to harvest email addresses from Internet Service Providers, mail services such as Hotmail, and private corporations. In the attack, software opens up a connection to a mail server and sends millions of delivery attempt requests to email addresses on the server, such as timsmith@hotmail.com, tomsmith@hotmail.com, tom1smith@hotmail.com, and so on.



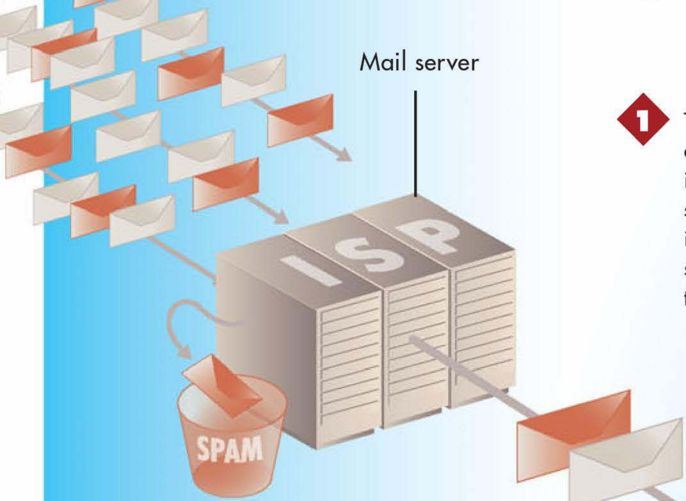

**6** Because the software is only guessing at the addresses, most of the addresses are invalid, and the server responds with a Simple Mail Transfer Protocol (SMTP) 550 error message, which means that the address is not a valid one.

**7** When the address being sent is a real one, the server responds with a message that the address is valid and real. The spammer compiles all of these into a spam list. Even though a very low percent of addresses are valid, if the spammer has sent millions of requests, they will get tens of thousands, or more, of valid email addresses.

### The Speed at Which Spam Spreads

How effective are spammers in harvesting email addresses from public locations? According to an investigation by the Federal Trade Commission and several law enforcement agencies, they are remarkably efficient. The commission and agencies posted 250 fresh, new email addresses in 175 locations on the Internet to see how much spam each received. The addresses were posted on web pages, dating services, chat rooms, message boards, Usenet newsgroups, and other locations. In the six weeks after posting, the addresses had received 3,349 pieces of spam. Eighty six percent of addresses posted to web pages drew spam, and an equal percent of addresses posted on newsgroups drew spam. And chat rooms were possibly the biggest spam magnets of all: One address used in a chat room received spam a mere nine minutes after it was first posted.

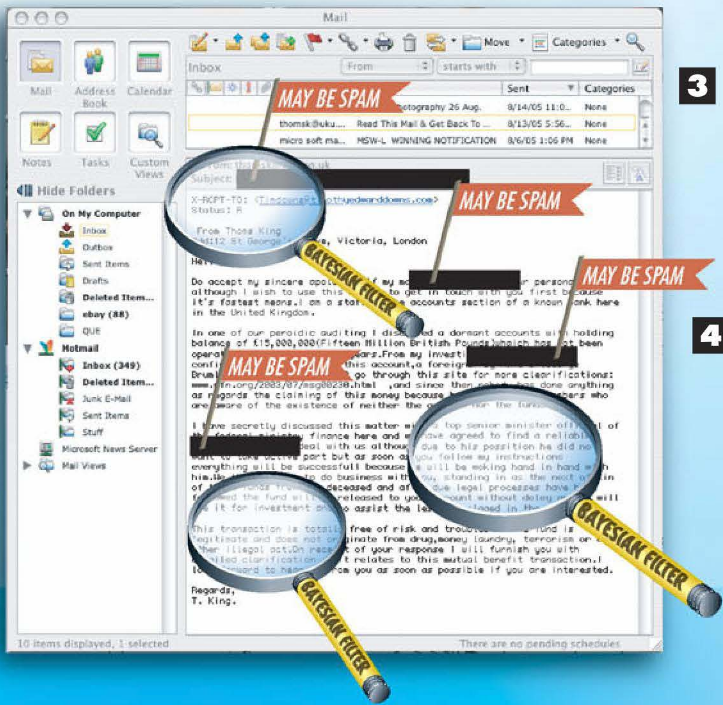
# How Antispam Software Sniffs Out Phony Email




**1** There are many types of antispam software. Some reside on a mail server and delete spam before it reaches people's PCs. Another common type runs inside a person's email program and routes suspected spam to a special spam folder, where the mail can be examined and deleted. Most ISPs include antispam software, but that in itself isn't enough to stop the deluge of spam from reaching people, so most people also run antispam software on their PCs.

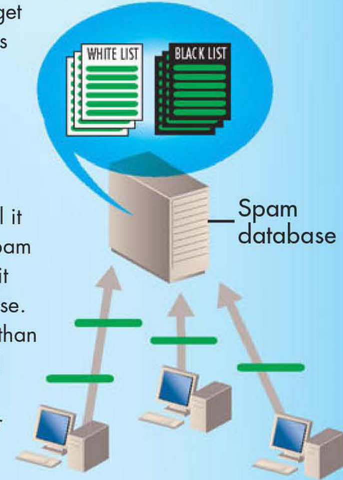


**2** Blacklists are a common way of fighting spam. A blacklist is a list of email addresses and Internet domains known to be spammers. When antispam software comes across an address on the blacklist, it flags it as spam.




**3** Antispam software also uses content filtering, in which the software examines the body and subject line of an email message and looks for specific words that indicate spam. The software contains a database of terms and phrases that spammers often use, such as *Viagra*. So if the software comes across a subject line such as *Herbal Viagra!*, it would consider the message spam.

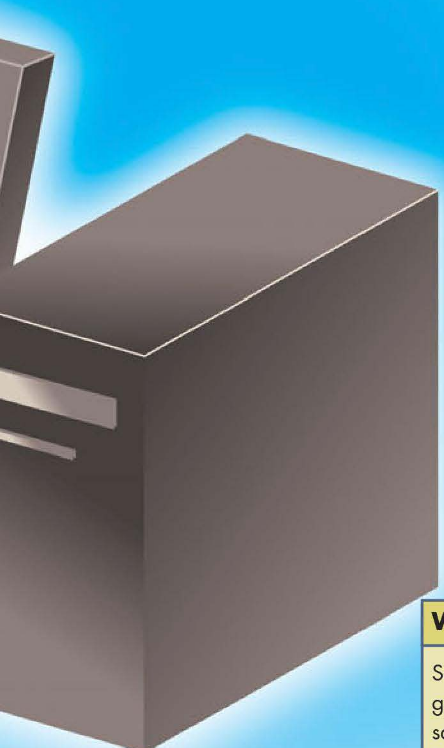
**4** A Bayesian filter might be the most powerful antispam technique of all. A Bayesian filter analyzes the actual content of a message, compares the content to a database of spam characteristics, and calculates the probability that the message is spam. All messages above a certain threshold are considered spam, and messages below that threshold are not considered spam. You can tune the filter to change the threshold level, depending on whether you want to be more aggressive or less aggressive in flagging spam. Being more aggressive means you'll get less spam, but also more **false positives**, messages that are not actually spam. Being less aggressive means that more spam will get through, but you'll get fewer false positives. When you use a Bayesian filter, as you get mail, you flag certain messages as spam and others as not being spam. So the more use you a Bayesian filter, the more effective it becomes because as you tell it what is spam and what isn't, it adds that information to its database. Bayesian filters are more effective than content filters that only block email with certain words or phrases because spammers can easily alter the spelling of words.





**5** Whitelists can be used to tell antispam software that certain addresses are valid, and so should always be let through by antispam software. For example, some antispam software examines a person's contact list and automatically adds all those addresses to a whitelist. Additionally, you can add addresses to a whitelist, telling the antispam software to let certain addresses through.

**6** Some software uses peer-to-peer technology to fight spam. Everyone who uses the antispam software flags certain messages as spam. This information is sent to a central server, which compiles blacklists and whitelists. The software on every person's computers is then updated by the central server.

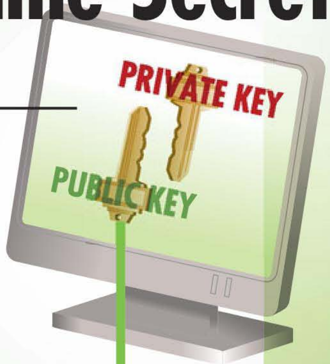


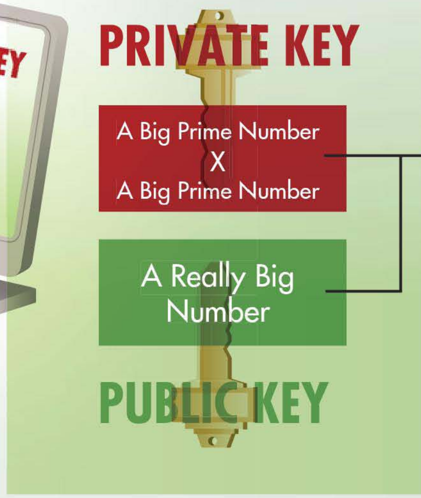
### What Is Spammerwocky?

Some spam includes random words and lines of gibberish, such as "inexorable lie stone liver conclude grandma trickster." This technique, called spammerwocky, tries to fool spam filters into believing the message is not spam. The spammer hopes that the inclusion of random words will not be construed as spam by Bayesian filters. But spam-killing software has caught on and includes methods for detecting spammerwocky.

# How Prime Numbers Protect Prime Secrets

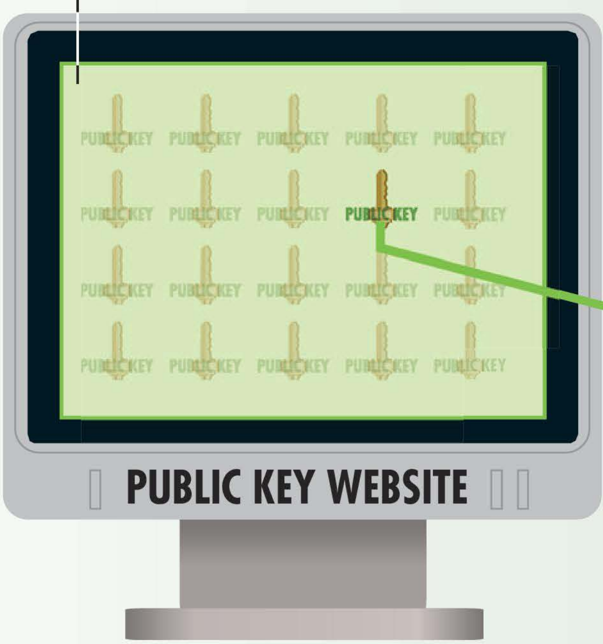
**1** A person who wants to be sure information she exchanges over the Internet is not read by others and is not a forgery uses **encryption** software to create two **keys**.



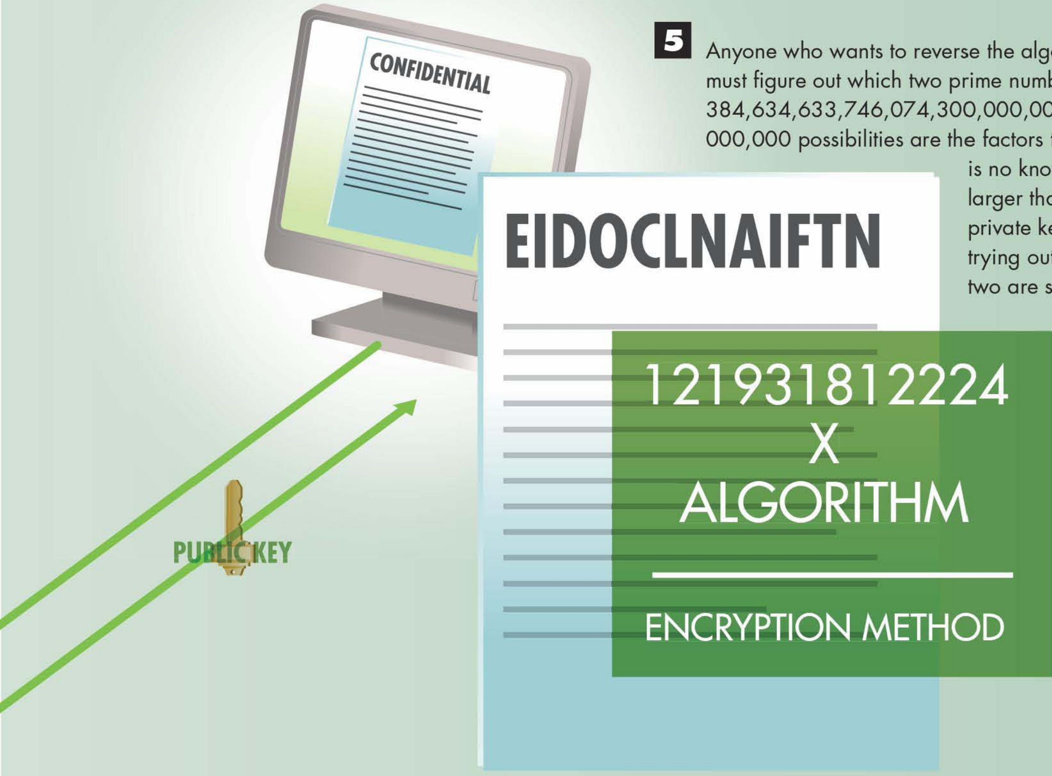




**2** The software typically multiplies two **prime numbers**—numbers that can be evenly divided only by itself and the number 1. The software uses up to 128 bits to record those numbers. With that many bits, there are a possible 3,402,823,669,209,384,634,633,746,074,300,000,000,000,000,000,000,000,000,000,000,000 different combinations. If, say, both of those numbers have 75 digits, their product will consist of 150 digits. Those two prime numbers become a **private key**. The person who creates them is the only one who possesses the private key.

**3** The person then posts the product of multiplying those keys somewhere it can be read by anyone—a printed or online directory. That number becomes the **public key**.

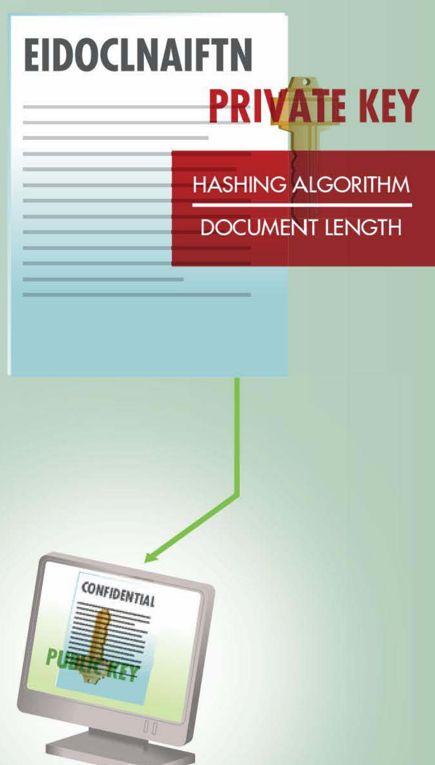




**4** Another person who wants to send the first person a confidential document encrypts the file using the public key as a variable in the **algorithm** used by the software. An algorithm is a fixed set of operations that change data in a way that makes the original document incomprehensible. A simple example of an algorithm is "shift one letter to the right," so that HAL becomes IBM. The key to decrypting it would be "shift one letter to left."

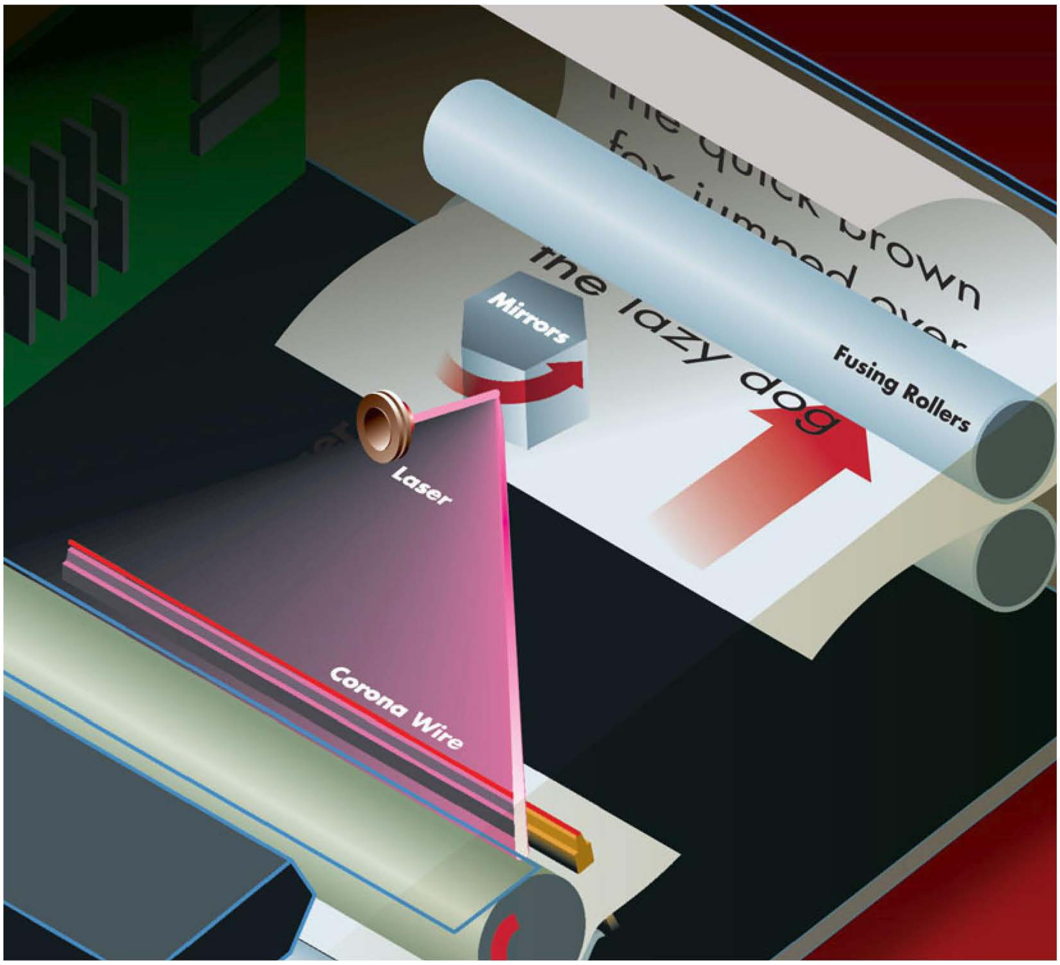



**5** Anyone who wants to reverse the algorithm to restore the original message must figure out which two prime numbers out of the 3,402,823,669,209, 384,634,633,746,074,300,000,000,000,000,000,000,000,000,000, 000,000 possibilities are the factors that created the public key. Because there is no known formula for factoring large numbers larger than 80 digits, the only way to find out the private key is through brute-force computations—trying out every possible combination until the right two are stumbled upon. Using the most powerful computers available would take decades, if not centuries.

**6** Public key encryption is also used to create **digital signatures**. A digital signature is typically created by computing a **message digest** or **hash value**. These are numbers created when the contents of the document are run through a hashing algorithm. The resulting value is a mathematical summary of the document. The hash value is then encrypted using the private key.




**7** The recipient of the message uses the sender's public key to decrypt the hash values. The recipient runs the document through the same algorithm the sender did and, if the document is from whom it claims to be from, the two sets of hash values will match. If as little as a comma has been changed, or if the sender is impersonating someone else, the hash values won't match.




**8,000–3100 B.C.**
In Mesopotamia, tokens are used for accounting and recordkeeping.

**4000 B.C.**
Sumerians and other Near Eastern cultures begin using a system of picture writing that develops into cuneiform 1,000 years later.

**3500 B.C.**
In Sumer, pictographs of accounts are written on clay tablets.

**2600 B.C.**
Scribes are employed in Egypt.

**2400 B.C.**
In India, engraved seals identify the writer.

**2200 B.C.**
Date of oldest existing document written on papyrus.

**1700 B.C.**
Phoenicians use a 22-letter alphabet.

**1400 B.C.**
Oldest record of writing in China, on bones.

**1050 B.C.**
The Greeks come up with the novel idea of using signs for vowels as well as consonants, so any word can be written.

**775 B.C.**
Greeks develop a phonetic alphabet, written from left to right.

**500 B.C.**
Chinese scholars write on bamboo with reeds dipped in pigment.

**400 B.C.**
Chinese write on silk as well as wood and bamboo.

**200 B.C.**
Tipao gazettes are circulated to Chinese officials.

**200 B.C.**
Books are written on parchment and vellum.

P A R T

# 8

# How Printers Work

C H A P T E R S



**59 B.C.** Julius Caesar orders postings of *Acta Diurna.*

**105** T'sai Lun invents paper in China.

**175** Chinese classics are carved in stone that will later be used for rubbings.

**180** In China, an elementary zoetrope is invented.

**250** Paper use spreads to central Asia.

**350** In Egypt, a parchment book of Psalms is bound in wood covers.

**450** Ink on seals is stamped on paper in China. This is true printing.

**600** Books are printed in China.

**868** The Diamond sutra scroll, the oldest surviving document printed with blocks rather than copied by hand, is produced in China.

**875** Amazed travelers to China see toilet paper.

**950** Paper use spreads west to Spain.

**1000** Mayas in Yucatan, Mexico, make writing paper from tree bark.

**1035** Japanese use waste paper to make new paper.

**1049** Pi Sheng fabricates movable type using clay.

**1116** Chinese sew pages to make stitched books.

**1140** A crusader taken prisoner returns with papermaking art, according to a legend.

**1140** In Egypt, cloth is stripped from mummies to make paper.

*From the earliest days the hands of printers have wielded a great power, the magical power to reproduce words a thousandfold.*

—Helmut Presser

IN the early days of personal computers, when people thought it was simply a revolution rather than a fundamental change in our existence, someone came up with the idea that all this computerized data would lead to the "paperless office." We're entering the second century in which we will have been using computers, and more trees than ever are giving their lives to produce hard copies of everything from company budgets complete with full-color graphs to homemade greeting cards. Not only are we creating more printouts than ever before, but computer printing has turned into a fine art. The very essence of a whole new category of software—desktop publishing—is the accomplishment of better and better printed pages.

Whoever made that erroneous prediction about a paperless office missed an important fact. That person was probably thinking about how offices used paper in the age of the typewriter. Back then there wasn't much you could put on paper except black letters and numbers—most often in an efficient but drab typeface called Courier. If all those ugly memos and letters had been replaced by electronic mail, the world would not have suffered a great loss. But what forecasters didn't see is that software and printing technology would make possible fast, easy, graphic, and colorful hard copies of reports, newsletters, graphs, and, yes, company budgets and greeting cards, that even IBM's best Selectric could never come close to producing.

Speed and ease were the first improvements in printing. Whereas a simple typo on a typewriter might just be whited out or hand-corrected with a pen, today—because of the speed of printers—it's easier just to correct a mistake onscreen and print a fresh, flawless copy. Graphics

**1241** In Korea, metal type is used.

**1282** In Italy, watermarks are added to paper.

**1298** Marco Polo describes the use of paper money in China.

**1300** Wooden type is found in central Asia.

**1305** Taxis family begins private postal service in Europe.

**1309** Paper is used in England.

**1392** Koreans have a type foundry to produce bronze characters.

**1423** Europeans begin Chinese method of block printing.

**1450** A few newsletters begin circulating in Europe.

**1451** Johannes Gutenberg uses a press to print an old German poem.

**1452** Metal plates are used in printing.

**1454** The Gutenberg Bible is printed, the first European text printed using metal movable type.

**1490** Printing of books on paper becomes more common in Europe.

**1495** A paper mill is established in England.

**1500** By now, approximately 35,000 books have been printed, some 10 million copies.

**1545** Garamond designs his typeface.

**1550** Wallpaper is brought to Europe from China by traders.

**1565** The pencil comes out.

**1609** First regularly published newspaper appears in Germany.

**1631** The "Wicked Bible," as it was called, got its name from a misprint in the 14th line of the Ten Commandments.

were the next big advance. The day of the all-text document ended with the first software that *could* print even the crudest *line* graph on a dot-matrix printer. Now anything that's visual, from line art to a halftone photograph, can be printed on a standard office printer.

Today, color is the current frontier being conquered with office printers. The quality and speed of color printers is increasing as their cost is decreasing. Because they can double as black-and-white printers, you can expect to see the color printer increasingly the only printer in the office and home.


**Apple printer**

The first personal printers, such as this one for the Apple II, were impact dot-matrix, which created hard copy by pounding an ink-laden ribbon with small metal wires.

Paper hasn't disappeared from the office. Instead, it's taken on a whole new importance. And the lowly printer that used to turn out crude approximations of characters is now one of the most important components of a computer system.

We'll start by looking at the dot-matrix impact printer. Black-and-white impact printers have been pushed out of the office and home markets by cheap, colorful ink-jet printers. With each new edition, I'm tempted to take it out of this forward-looking book, but understanding the dot-matrix printer is a good foundation for understanding how other, more sophisticated printers work. Other printers make their dots without the miniature violence an impact printer commits on a

**1639** First printing press in the American colonies.

**1689** Newspapers are printed, at first as unfolded "broadsides."

**1696** By now England has 100 paper mills.

**1710** German engraver Le Blon develops three-color printing.

**1714** Henry Mill receives patent in England for a typewriter.

**1719** Reaumur proposes using wood to make paper.

**1725** Scottish printer develops stereotyping system.

**1770** The eraser is introduced.

**1774** Swedish chemist invents a future paper whitener.

**1784** French book is made, without rags, from vegetation.

**1798** Senefelder in Germany invents lithography.

**1799** Robert in France invents a paper-making machine.

**1800** Paper can be made from vegetable fibers instead of rags.

**1803** Fourdrinier continuous web papermaking machine is invented.

**1804** In Germany, lithography is invented.

**1806** Carbon paper is invented.

**1808** Turri of Italy builds a typewriter for a blind contessa.

**1867** Christopher Sholes, a Milwaukee newspaper editor, invents the typewriter.

**1901** First electric typewriter, the Blickensderfer.

**1902** Etched zinc engravings start to replace hand-cut wood blocks.

**Epson printer**

Today, the standard for printing is the color inkjet that produces full-color hard copies in seconds.

ribbon smeared with soot. But any printer, whether dot-matrix, ink-jet, laser, dye sublimation, or solid ink, accomplishes essentially the same task: It creates a pattern of dots on a sheet of paper. The dots can be sized differently or composed of different inks that are transferred to the paper by different means, but all the images for text and graphics are made up of dots. The smaller the dots, the more attractive the printout.

This is a fundamental change in computer printing from all printing that's gone before it since Gutenberg. The freedom that Gutenberg's movable type brought also created restrictions. Printing was done with fully formed characters, letters, and numbers. The dot matrix changed that. With it, a printer could print anything that could be made up of many tiny dots, which as we see today, is anything. It led directly to what have become the standards for home and office printing—ink-jets for the home and laser printers for the office. Like an impact dot-matrix printer, they create images with dots. It's just that the process is hidden and silent relative to the pounding whine of an impact printer.

**1902** The first teleprinters.

**1904** A photograph is transmitted by wire in Germany.

**1906** In Britain, a new process colors books cheaply.

**1923** A picture, broken into dots, is sent by wire.

**1924** Low-tech achievement: notebooks get spiral bindings.

**1928** The teletype machine makes its debut.

**1929** Bell Lab transmits stills in color by mechanical scanning.

**1934** The Associated Press starts a wirephoto service.

**1935** The Penguin paperback book sells for the price of 10 cigarettes.

**1942** Kodacolor process produces the color print.

**1904** The comic book is invented.

**1905** Photography, printing, and post combine in the year's craze, picture postcards.

**1917** Teletypewriters appear, allowing point-to-point printed communications. Today's TTY terminal emulation standard dates back to this device.

**1926** The Book-of-the-Month Club is founded.

**1929** Telegraph ticker sends 500 characters per minute.

**1932** The Times of London uses its new Times Roman typeface.

**1935** IBM's electric typewriter comes off the assembly line.

**1938** Two brothers named Biro invent the ballpoint pen in Argentina.

**1950** Changeable typewriter typefaces are in use.

# KEY CONCEPTS

**Ben Day dots** The individual dots of ink that make up a printed graphic.

**bitmapped font** Characters created by using records of where each and every dot of ink is placed in a dot matrix. Sometimes referred to as raster fonts, they are most often used in impact and low-end printers. The use of bitmapped fonts is decreasing because of the versatility of outline fonts used in Windows.

**CYMK** Cyan, yellow, magenta, and black (K)—the four colors most often used in color printing.

**dithering** A process in which the frequency and placement of ink are used to create shades of gray and hues of color.

**dot matrix** The grid of horizontal and vertical dots that make up all the possible dots—most often as many as 900,000—that could be included in the bitmap of a single character. Don't confuse dot matrix with impact printers. An ink-jet printer also creates bitmapped letters using a matrix.

**DPI** Dots per inch, the number of ink dots a printer lays down in a single row one inch long. The higher the dpi, the better the resolution.

**font/typeface** A typeface is a design for the alphabet distinguished by its use of such elements as serifs, boldness, and shape. Times Roman, Helvetica, and Courier are typefaces. So a font is a typeface of a particular size and a particular variation, such as italic. Courier 12-point bold and Courier 12-point italic are different fonts in the Courier typeface family.

**impact printer** A printer that uses a quick blow to press ink from a ribbon onto paper.

**ink-jet printers** Ink-jet printers use tiny nozzles to spray precise patterns of black and colored ink on plain or special-purpose papers.

**laser printers** A printer that uses the energy from a rapidly flashing laser to create the images of the page on a special drum. Through various exchanges of positive and negative static electricity charges, a laser printer transfers black, or colored, powders from the drum to a sheet of paper.

**multifunction** Multifunction office machines provide printer, scanner, copier, and fax machine capabilities. They are less expensive than buying separate machines and take up less office space. Also called "all-in-one" printers. They may use either ink-jet or laser printing technology.

**page description language (PDL)** A software language used with printers to control complex and sophisticated print jobs. PostScript and TrueType are the two most common page description languages.

**point** 1/72 of an inch, a traditional measure for typefaces.

**PPM** Pages per minute, a measurement of the speed at which a printer completes pages. Document contents, graphics, and the size of a document influences ppm speeds.

**print head** The mechanism that actually transfers ink from the printer to paper.

**resolution** The quality of the text and images on hard copy are dependent largely on the resolution of the printer, which is determined by the number of dots of ink it would take to make a one-inch line. 300 dpi (dots per inch) is most common, although 600 and even 1,200 dpi printers are becoming more commonplace.

**toner** Powdered plastic used in laser printers in place of ink.

**1957** First book to be entirely photo typeset is offset-printed.

**1959** Xerox manufactures a plain paper copier.

**1960** Fibertip pen is introduced.

**1961** Letraset makes headlines simple.

**1961** IBM introduces the "golf ball" typewriter.

**1966** Xerox sells the Telecopier, a fax machine.

**1966** Linotron can produce 1,000 characters per second.

**1968** Rand Laboratories develops the Rand Tablet, which translates handwriting into typed text.

**1969** Gary Starkweather, at Xerox's research facility in Webster, New York, demonstrates using a laser beam with the xerography process to create a laser printer.

**1971** The dotmatrix printer is introduced.

**1973** IBM's Selectric typewriter is now "self-correcting."

**1975** The laser printer is invented.

**1977** Canon researcher invents inkjet technology when he accidentally touches an ink-filled syringe with a hot soldering iron. The heat forced a drop of ink out of the needle. The ink-jet printer is invented.

**1978** Electronic typewriters go on sale.

**1983** Lasers and plastics improve newspaper production.

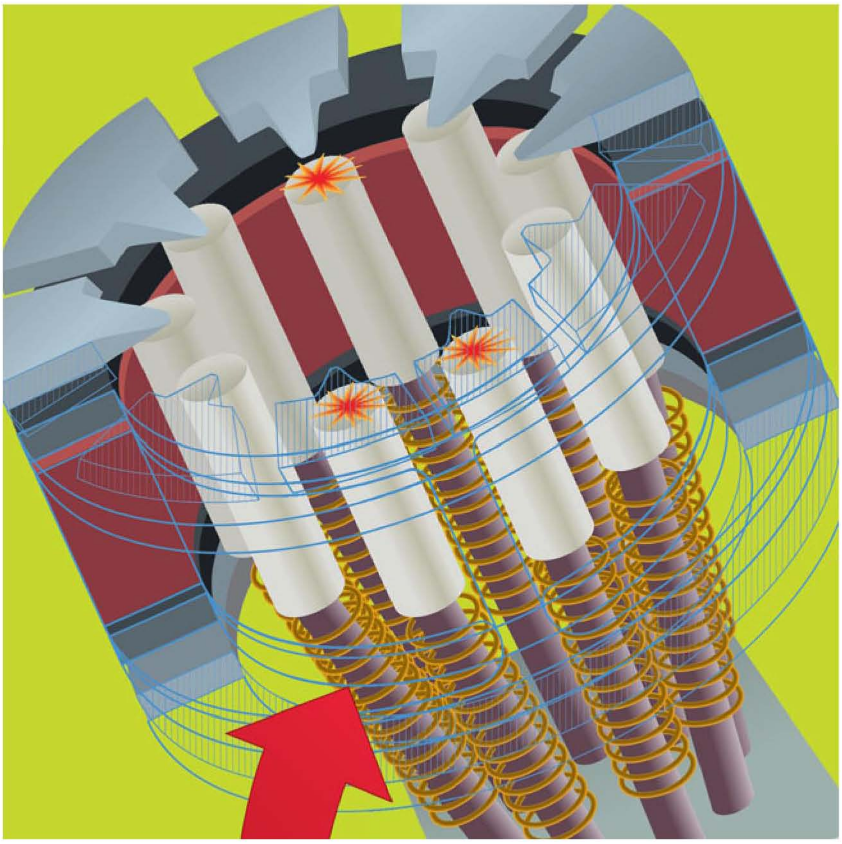
**1984** Hewlett Packard introduces the laser printer.

**1990** IBM sells Selectric, a sign of the typewriter's passing.

CHAPTER

# 32

# How Black-and-White Printing Works



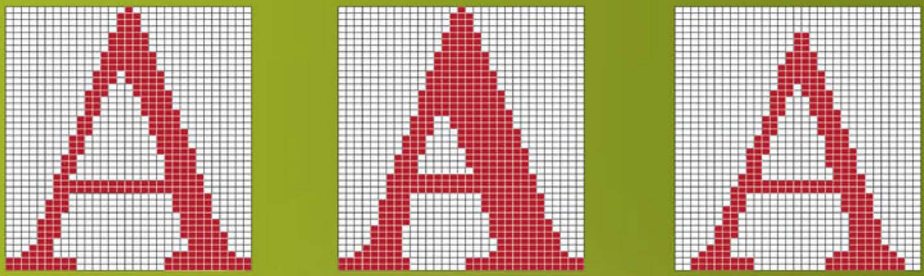
**ALL** computer printing these days is based on the dot matrix. Whether it's a laser printer going through an intricate ballet of movement and time or an ink-jet printer spitting dots of color on paper, the printer is limited to producing dots. Thousands of dots on a single page, but still dots.

Regardless of how the dots are created, there must be a common method for determining where to place the dots. The most common schemes are bitmaps and outline fonts. Bitmapped fonts come in predefined sizes and weights. Outline fonts can, on the fly, be scaled and given special attributes, such as boldfacing and underlining. Each method has its advantages and disadvantages, depending on what type of output you want.

Bitmapped images are the computer's equivalent of Gutenberg's type. Bitmaps generally are limited to text and are a fast way to produce a printed page that uses only a few type fonts. If the hard copy is to include a graphic image in addition to bitmapped text, then, to create the graphic, your software must be able to send the printer instructions that it will understand.

Outline, or vector fonts, are used with a page description language, such as Adobe Postscript or Microsoft TrueType, that treats everything on a page—even text—as a graphic. The text and graphics used by the software are converted to a series of commands that the printer's page description language uses to determine where each dot is to be placed on a page. Page description languages are no longer so much slower than matrix printers. Outline fonts are more versatile at producing different sizes of type with different attributes or special effects, and they create more attractive results. It is the printing triumph of the bit.

# How Printers Make Cookie-Cutter Text

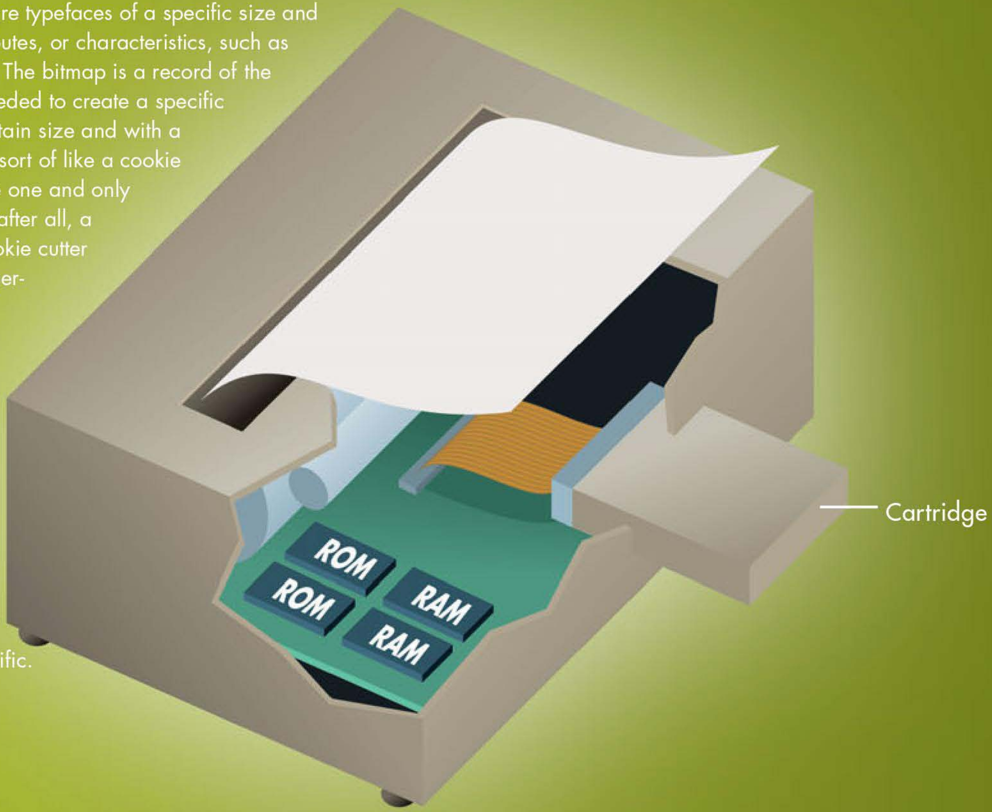


36-pt. medium

36-pt. bold

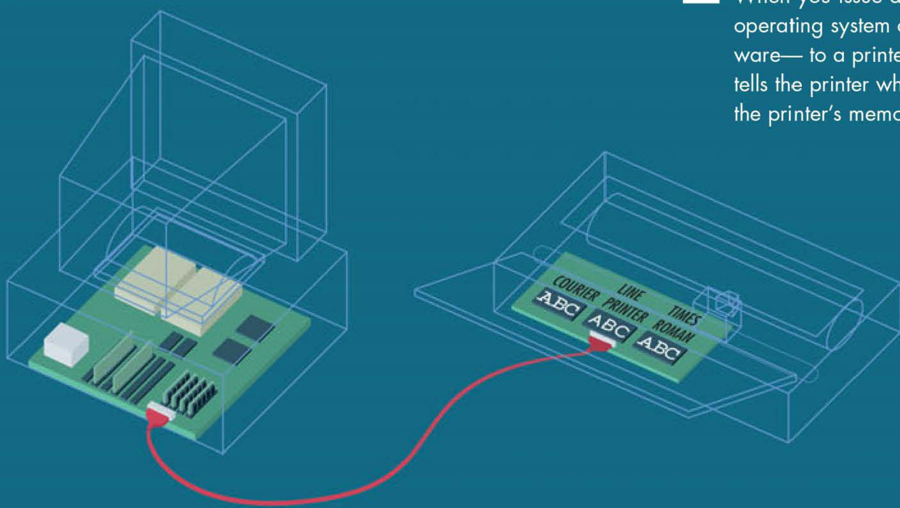
30-pt. medium

**1** Bitmapped fonts are typefaces of a specific size and with specific attributes, or characteristics, such as boldface or italic. The bitmap is a record of the pattern of dots needed to create a specific character in a certain size and with a certain attribute—sort of like a cookie cutter. It can make one and only one kind of type; after all, a Christmas tree cookie cutter can't make a gingerbread boy. The bitmaps for a 36-point Times Roman medium capital A, for a 36-point Times Roman boldface capital A, and for a 30-point Times Roman medium capital A are all different and specific.

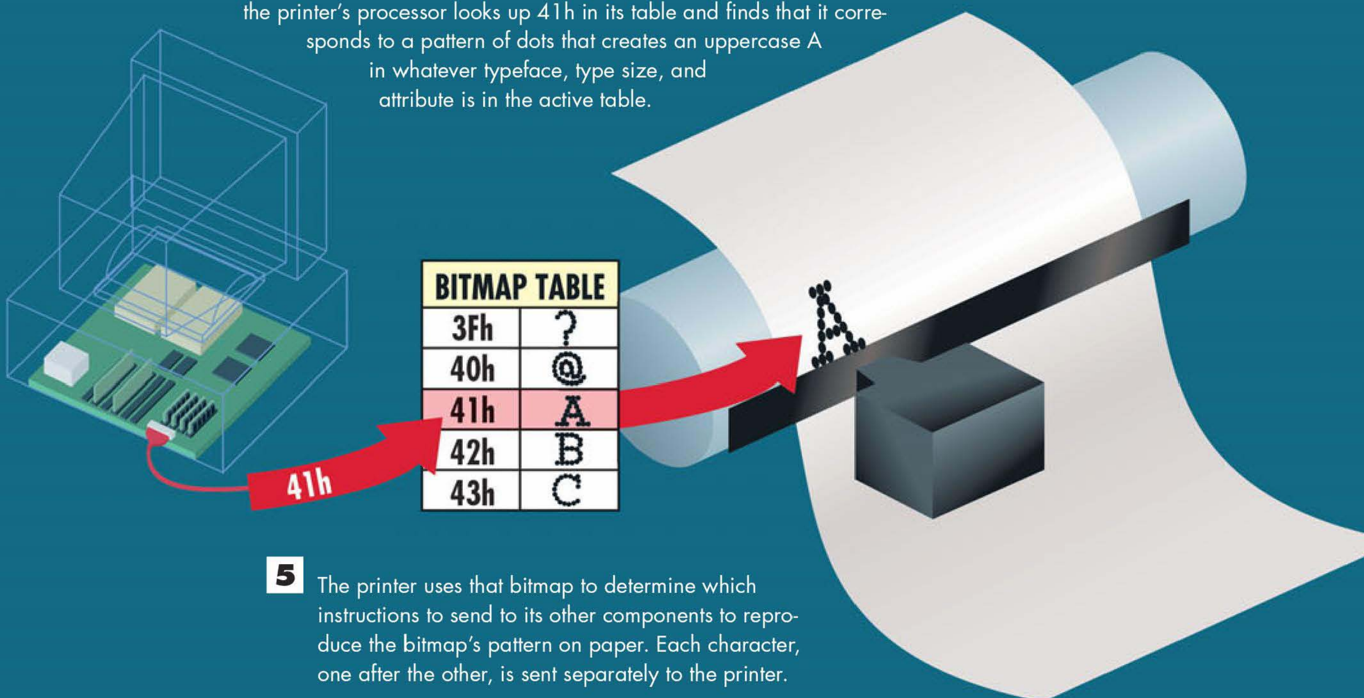



**2** Most printers come with a few bitmapped fonts—usually Courier and Line Printer—in both normal and boldface varieties as part of their permanent memory (ROM). In addition, many printers have random access memory (RAM) to which your computer can send bitmaps for other fonts. You also can add more bitmapped fonts in the form of plug-in cartridges that some laser printers use.

**3** When you issue a print command—either from your operating system or from within your application software— to a printer using bitmapped fonts, your PC first tells the printer which of the bitmap tables contained in the printer's memory it should use.

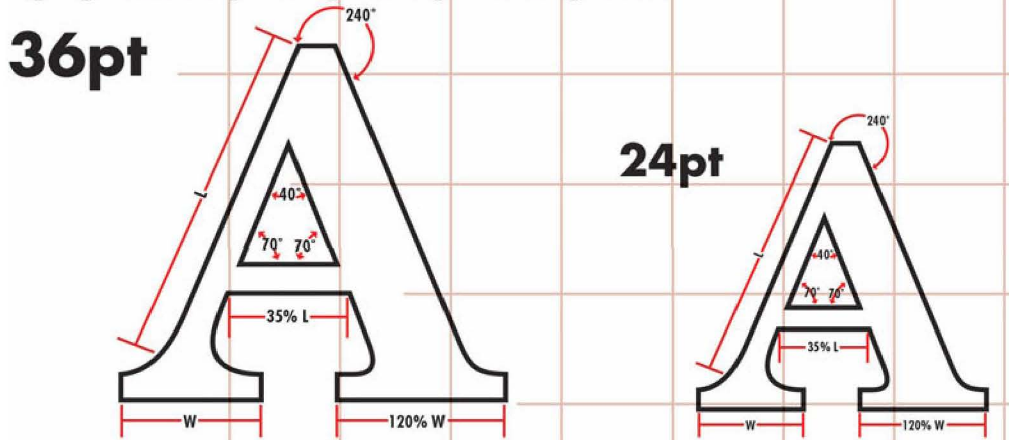



**4** Then, for each letter, punctuation mark, or paper movement—such as a tab or carriage return—that the software wants the printer to create, the PC sends an ASCII code. ASCII codes consist of hexadecimal numbers that are matched against the table of bitmaps. (Hexadecimal numbers have a base of 16—1, 2, 3, 4, 5, 6, 7, 8, 9, 0, A, B, C, D, E, F—instead of the base 10 used by decimal numbers.) If, for example, the hexadecimal number 41 (65 decimal) is sent to the printer, the printer's processor looks up 41h in its table and finds that it corresponds to a pattern of dots that creates an uppercase A in whatever typeface, type size, and attribute is in the active table.

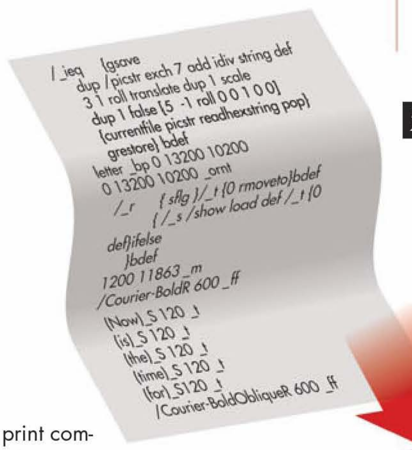



**5** The printer uses that bitmap to determine which instructions to send to its other components to reproduce the bitmap's pattern on paper. Each character, one after the other, is sent separately to the printer.

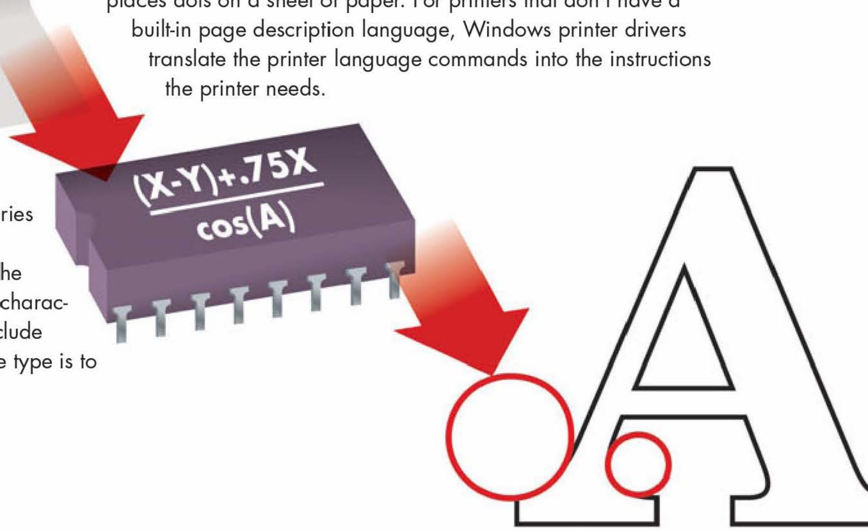
# How Outline Fonts Work




**1** Outline fonts, unlike bitmapped fonts, are not limited to specific sizes and attributes of a typeface. Instead, they consist of mathematical descriptions of each character and punctuation mark in a typeface. They are called outline fonts because the outline of a Times Roman 36-point capital A is proportionally the same as that of a 24-point Times Roman capital A.

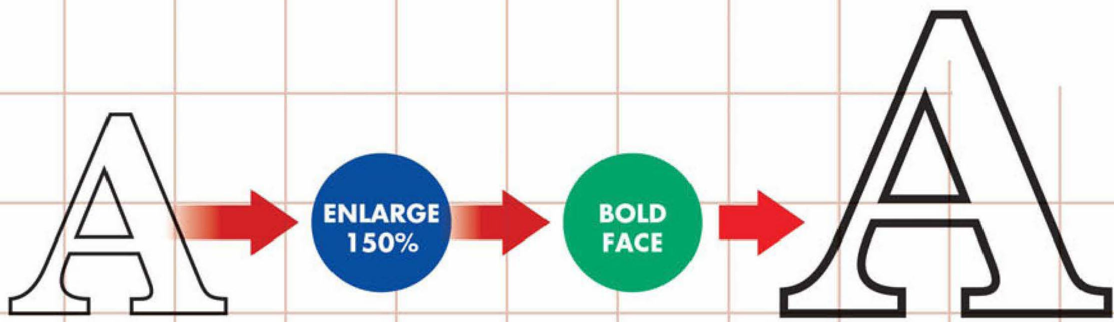



**2** Some printers come with a page description language, most commonly PostScript or Hewlett-Packard Printer Command Language, in **firmware**—a computer program contained on a microchip. The language can translate outline font commands from your PC's software into the instructions the printer needs to control where it places dots on a sheet of paper. For printers that don't have a built-in page description language, Windows printer drivers translate the printer language commands into the instructions the printer needs.




**3** When you issue a print command from your application software using outline fonts, your application sends a series of commands the page description language interprets through a set of algorithms, or mathematical formulas. The algorithms describe the lines and arcs that make up the characters in a typeface. The algorithms for some typefaces include hints, special alterations to the details of the outline if the type is to be either extremely big or extremely small.

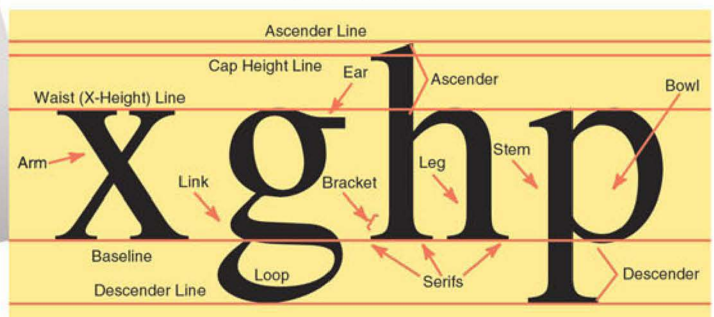
**4** The commands insert variable values into the formulas to change the size or attributes of the outline font. The results are commands to the printer that say, in effect, "Create a horizontal line 3 points wide, which begins 60 points from the bottom and 20 points to the right." The page description language turns on all the bits that fall inside the outline of the letter—unless the font includes some special shading effect within the outline.



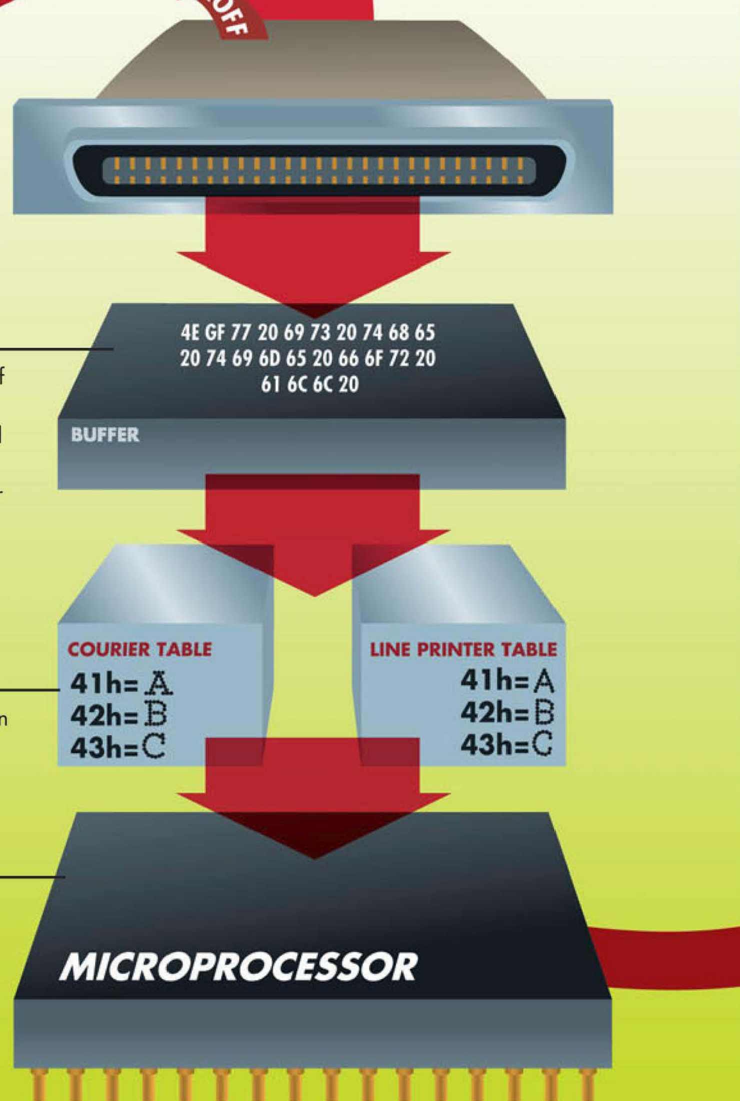


**5** Instead of sending the individual commands for each character in a document, the page description language sends instructions to the printing mechanism that produces the page as a whole. Under this scheme, the page essentially is one large graphic image that might also happen to contain text; text and graphics are treated the same here. Treating a page as a graphic rather than as a series of characters generally makes producing text slower with a page description language than with a bitmap.

Parts of a Letter



# How the First Printers Created Text

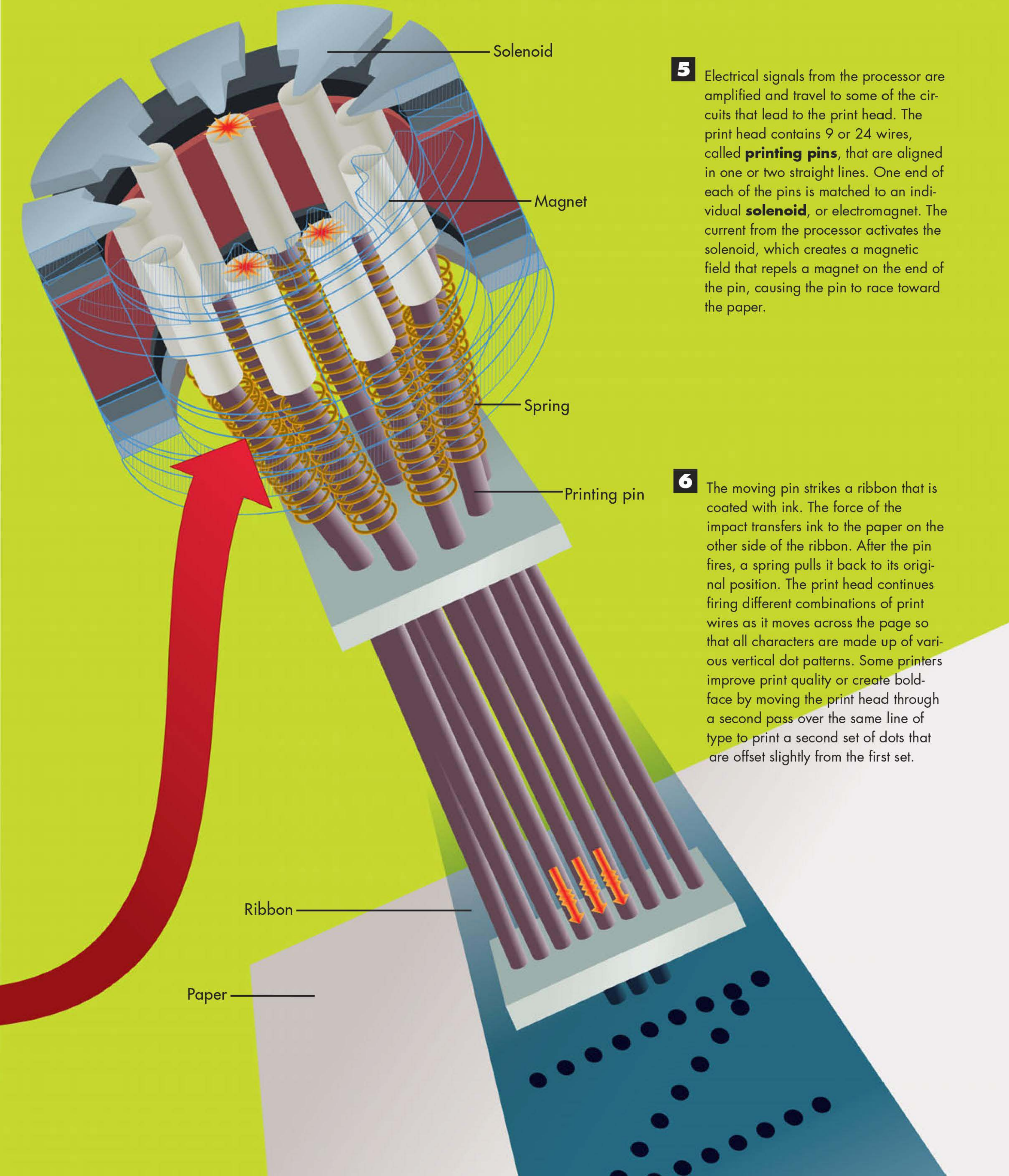




**1** Your PC sends a series of hexadecimal ASCII codes to an **impact printer**, also often referred to as a **dot-matrix printer**. The codes represent characters, punctuation marks, and printer movements such as tabs, carriage returns, line feeds, and form feeds that control the position of the print head in relation to the paper.

**2** The ASCII codes are stored in a **buffer**, which is a special section of the printer's random access memory (RAM). Because it usually takes longer for an impact printer to print characters than it takes a PC and software to send those characters to the printer, the buffer helps free up the PC to perform other functions during printing. When the buffer gets full, the printer sends an XOFF control code to the computer to tell it to suspend its stream of data. When the buffer frees up space by sending some of the characters to its processor, the printer sends an XON code to the PC, which resumes sending data.

**3** Among other codes are commands that tell the printer to use a certain font's **bitmap table**, which is contained in the printer's read-only memory chips. That table tells the printer the pattern of dots that it should use to create the characters represented by the ASCII codes.

**4** The printer's processor takes the information, which the bitmap table provides, for an entire line of type and calculates the most efficient path for the print head to travel. Some lines might be printed from right to left. The processor sends the signals that fire the pins in the print head, and it also controls the movements of the print head and platen.

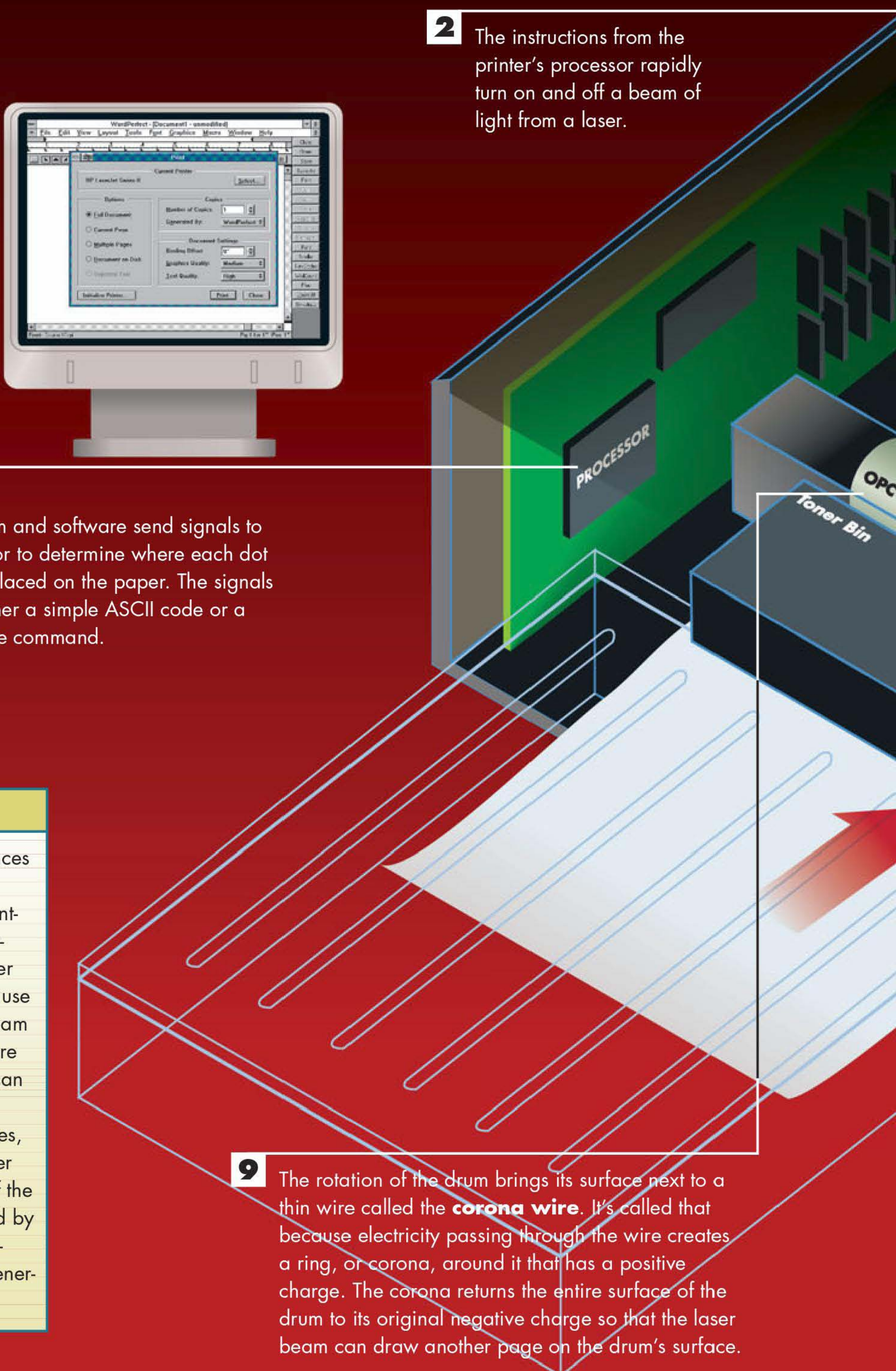



**5** Electrical signals from the processor are amplified and travel to some of the circuits that lead to the print head. The print head contains 9 or 24 wires, called **printing pins**, that are aligned in one or two straight lines. One end of each of the pins is matched to an individual **solenoid**, or electromagnet. The current from the processor activates the solenoid, which creates a magnetic field that repels a magnet on the end of the pin, causing the pin to race toward the paper.

**6** The moving pin strikes a ribbon that is coated with ink. The force of the impact transfers ink to the paper on the other side of the ribbon. After the pin fires, a spring pulls it back to its original position. The print head continues firing different combinations of print wires as it moves across the page so that all characters are made up of various vertical dot patterns. Some printers improve print quality or create boldface by moving the print head through a second pass over the same line of type to print a second set of dots that are offset slightly from the first set.

# How a Laser Printer Works

**2** The instructions from the printer's processor rapidly turn on and off a beam of light from a laser.

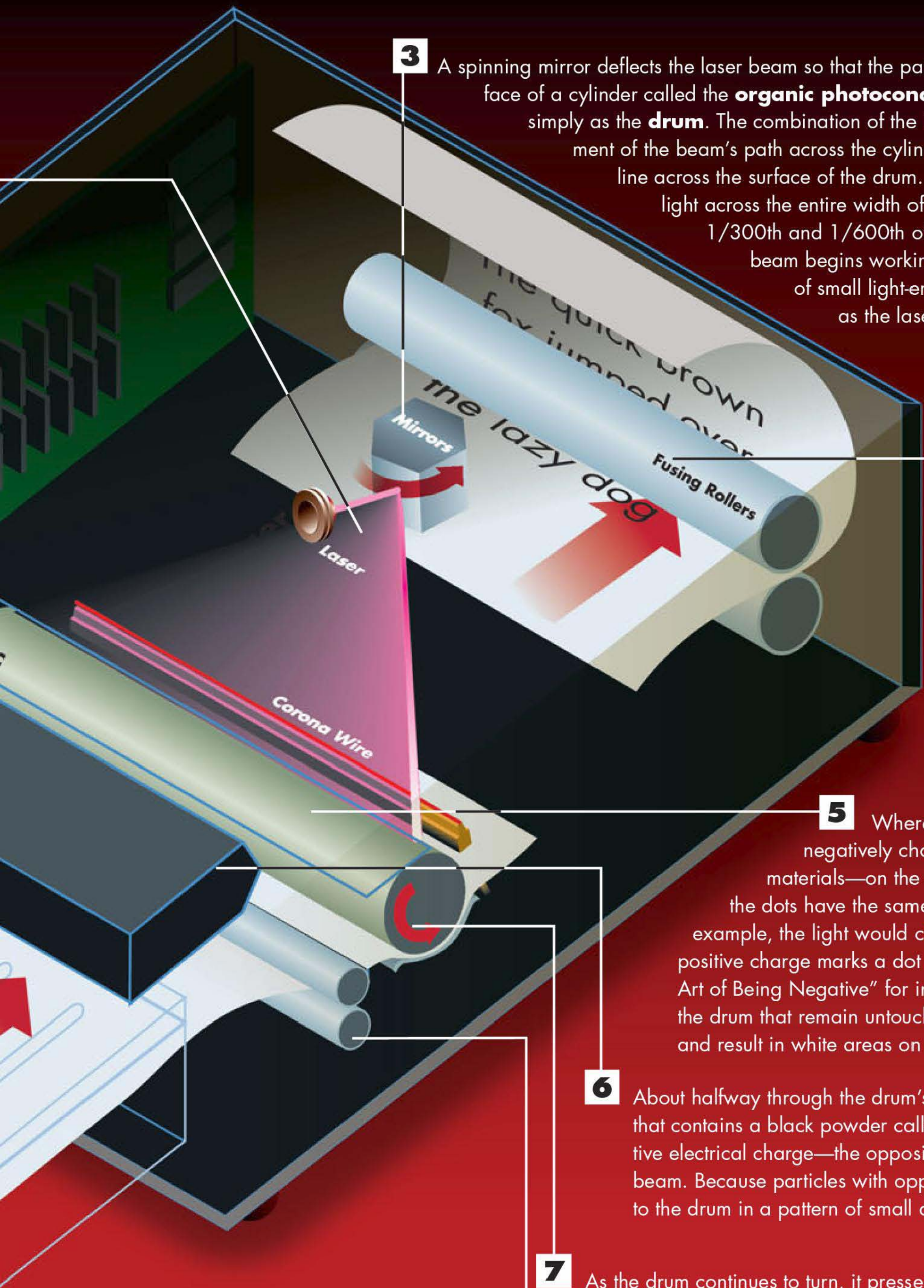



**1** Your PC's operating system and software send signals to the laser printer's processor to determine where each dot of printing toner is to be placed on the paper. The signals are one of two types—either a simple ASCII code or a page description language command.

### The Art of Being Negative

In this description, the electrical charges in all instances can be reversed and the result would be much the same. The method described here is true of most printers that use the Canon print engine, such as Hewlett-Packard models, which are the standard among laser printers. This approach is called **write-black** because every dot etched on the printer drum by the laser beam marks a place that will be black on the printout. There is, however, an alternative way that a laser printer can work, and that way produces noticeably different results. The other method, used by Ricoh print engines, is called **write-white** because everywhere the laser beam strikes, it creates a charge the same as that of the toner—the toner is attracted to the areas not affected by the beam of light. Write-white printers generally produce darker black areas, and write-black printers generally produce finer details.

**9** The rotation of the drum brings its surface next to a thin wire called the **corona wire**. It's called that because electricity passing through the wire creates a ring, or corona, around it that has a positive charge. The corona returns the entire surface of the drum to its original negative charge so that the laser beam can draw another page on the drum's surface.




**3** A spinning mirror deflects the laser beam so that the path of the beam is a horizontal line across the surface of a cylinder called the **organic photoconducting cartridge (OPC)**, usually referred to simply as the **drum**. The combination of the laser beam being turned on and off and the movement of the beam's path across the cylinder results in many tiny points of light hitting in a line across the surface of the drum. When the laser has finished flashing points of light across the entire width of the OPC, the drum rotates—usually between 1/300th and 1/600th of an inch in most laser printers—and the laser beam begins working on the next line of dots. Some printers use rows of small light-emitting diodes (LEDs) that serve the same function as the laser light with fewer moving parts.

**4** Another set of rollers pulls the paper through a part of the print engine called the **fusing system**. There, pressure and heat bind the toner permanently to the paper by melting and pressing a wax that is part of the toner. The heat from the fusing system is what causes paper fresh from a laser printer to be warm. The paper train pushes the paper out of the printer, usually with the printed side down so that pages end up in the output tray in the correct order.

**5** Where each point of light strikes the drum, it causes a negatively charged film—usually made of zinc oxide and other materials—on the surface of the drum to change its charge so that the dots have the same electrical charge as the sheet of paper. In this example, the light would change the charge from negative to positive. Each positive charge marks a dot that eventually will print black on paper. (See "The Art of Being Negative" for information about write-white printers.) The areas of the drum that remain untouched by the laser beam retain their negative charge and result in white areas on the hard copy.

**6** About halfway through the drum's rotation, the OPC comes into contact with a bin that contains a black powder called **toner**. The toner in this example has a negative electrical charge—the opposite of the charges created on the drum by the laser beam. Because particles with opposite static charges attract each other, toner sticks to the drum in a pattern of small dots wherever the laser beam created a charge.

**7** As the drum continues to turn, it presses against the sheet of paper being fed along the paper train. Although the electrical charge on the paper is the same as the charge of the drum created by the laser beam, the paper's charge is stronger and pulls the toner off the drum and onto the paper.

**8** At the same time that the drum begins to rotate, a system of gears and rollers feeds a sheet of paper into the print engine along a path called the **paper train**. The paper train pulls the paper past an electrically charged wire that passes a static electrical charge to the paper.

CHAPTER

# 33 How Color Printing Works

**THERE** were two revolutions in computer printing in the twentieth century. One was the laser printer, which brings typeset-quality printing of text and graphics to the masses. The second was the development of inexpensive, fast, high-quality color printing.

The complexity of color printing, of course, means trade-offs. At the low-price end is the color ink-jet printer. It is in some ways a dot-matrix printer without the impact and with four times the colors. A color inkjet costs barely more than a black-and-white ink-jet. The visual detail approaches that of laser printers, in some printers surpassing it. But ink-jet technology is relatively slow, and you always have to fuss with cleaning and replacing the ink-filled print heads. Color ink-jets are the ideal printer for the home, where printing volume is small, a budget might be nonexistent, and the flash of color in a school report or a greeting card is worth the extra wait.

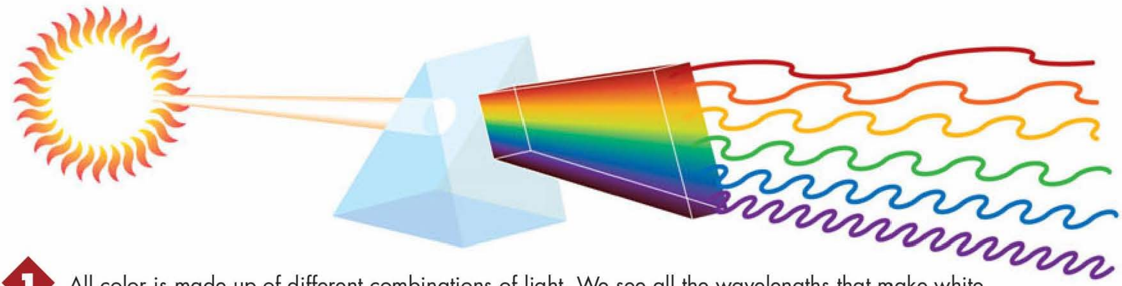
For the office, there are different color-printing solutions that match the budget of a small business or home office and solutions that give the most fussy graphic artists the speed, color-matching, and details they need to create professional results. The crucial difference among color printers is how they get ink on the paper. Because it involves four colors of ink to achieve full color printing, a printer must either make multiple passes over the same sheet of paper—as happens with some laser and thermal wax color printers—or it must manage to transfer all the colors more or less simultaneously, which is what happens with ink-jet, solid-ink, and other laser printers.

A common office color-printing device is the color **thermal printer**, which uses heat to transfer colored waxes from a wide ribbon to paper. The process provides vivid colors because the inks it uses don't bleed into each other or soak into specially coated paper. But its four-pass method is slow and wastes ink. The **color laser printer** provides the most precise detail, but is expensive because it requires four separate print engines that must each take their turn to apply colored toner to the page.

Two other color printing methods provide speed and photographic dazzle: **dye-sublimation**—also called **dye diffusion thermal transfer (D2T2)**—and **solid-ink**. By controlling not only how many dots of color they put on the page but the intensity of the dots, they produce continuous-tone printing. The result is virtually indistinguishable from a color photograph, even though its actual resolution might be no more than the 300 dots per inch of the old laser printer.

In this chapter, we'll look at how color printing tricks the eye into seeing hues and shades that aren't really there, and how ink-jet, photo, color laser, and solid-ink color printers work.

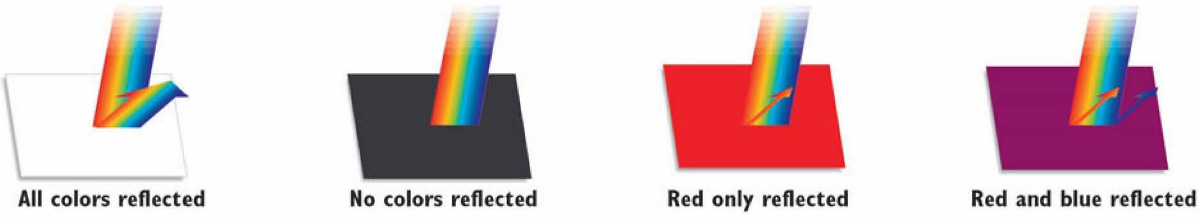
# How Printed Colors are Created



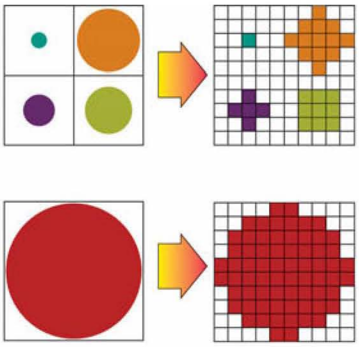
**1** All color is made up of different combinations of light. We see all the wavelengths that make white when we pass white light through a prism, breaking it into the spectrum. Although the spectrum is a continuous blending of colors, using only a few of those hues are necessary to produce color printing. Different mixtures of those primary colors re-create virtually any color from the spectrum by either adding colors or subtracting them.

**2** **Additive** color is used to create colors on televisions, computer monitors, and in movies. Three colors—red, green, and blue—are emitted to produce all other colors and white by adding various intensities of those primaries. Each time a color is added, it increases the number of colors the eye sees. If red, green, and blue are all added at their most saturated shades, the result is white.

**3** **Subtractive** color is the process used when light is reflected from colored pigments—rather than emitted as in additive color. Each added color absorbs (subtracts) more of the shades of the spectrum that makes up white light.

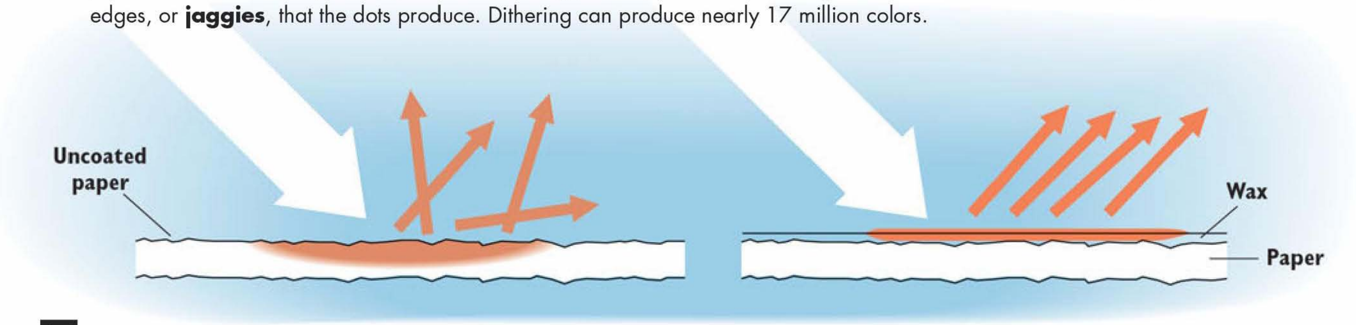



**4** Color printing uses four pigments: cyan (blue-green), yellow, magenta (purple-red), and black. This system is called **CYMK**. (K stands for black.) Some low-end color ink-jet printers save the cost of a black-ink print head by using equal portions of magenta, yellow, and cyan to produce black. But the resulting black lacks density, which is why better personal printers include a print head for black ink.



**5** All color printers use tiny dots of those four inks to create various shades of color on the page. Lighter shades are created by leaving dots of unprinted white. Some printers, such as dye-sublimation, control the size of the dots and produce continuous-tone images that rival photography. But most printers create dots that are essentially the same size no matter how much of a particular color is needed. The most common color printers create up to 300 dots of color per inch, for a total of 8 million dots per page. Many printers can create about 700 dpi, and a few printers can create up to 1,440 dpi.


**6** For all shades beyond the eight that are produced by overlaying the primaries, the printer generates a varied pattern of differently colored dots. For example, the printer uses a combination of one magenta dot to two of cyan to produce a deep purple. For most shades of color, the dots of ink are not printed on top of each other. Instead, they are offset slightly, a process called **dithering**. The eye accommodatingly blends the dots to form the desired shade as it hides the jagged edges, or **jaggies**, that the dots produce. Dithering can produce nearly 17 million colors.

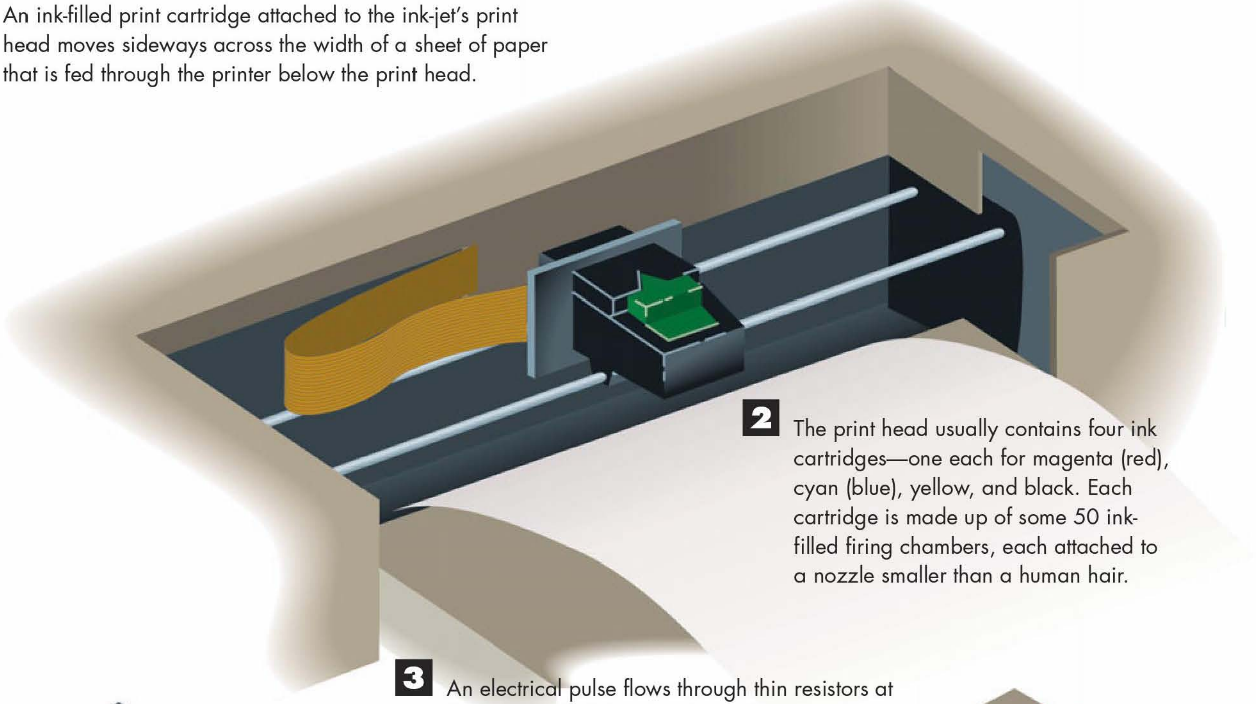



**7** The type of paper used in color printing affects the quality of the hard copy. Uncoated paper, the type used with most black-and-white office machines, has a rough surface that tends to scatter the light, reducing the brightness, and it tends to absorb ink, which slightly blurs the image.

Paper coated with a fine varnish or wax takes applications of ink more evenly so that the ink dries with a smooth surface that reflects more of the light hitting it. The coating also helps prevent the paper from absorbing the ink, producing a sharper image.

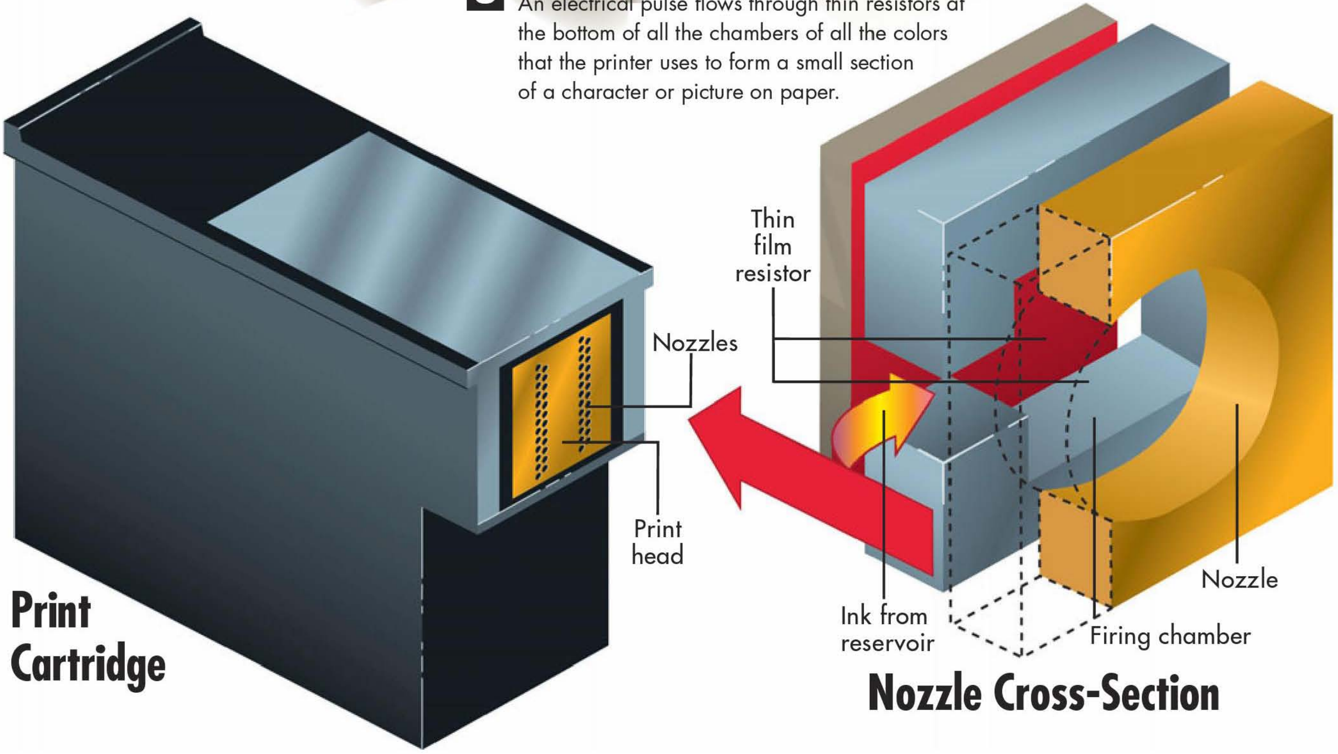
# How a Color Ink-jet Printer Works

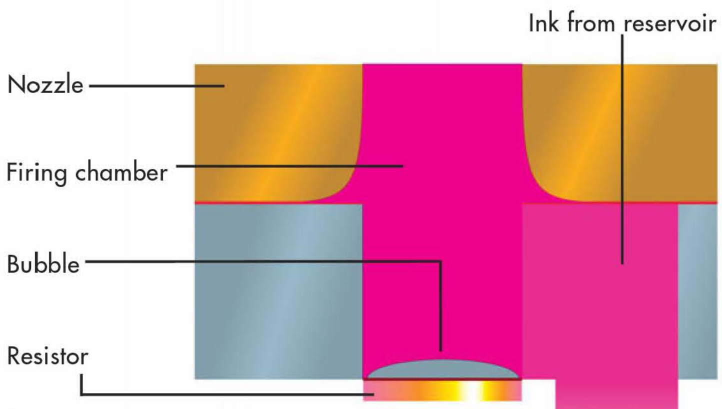
1. An ink-filled print cartridge attached to the ink-jet's print head moves sideways across the width of a sheet of paper that is fed through the printer below the print head.



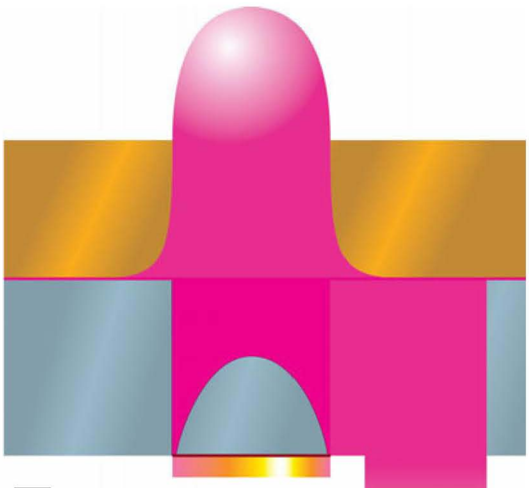
2. The print head usually contains four ink cartridges—one each for magenta (red), cyan (blue), yellow, and black. Each cartridge is made up of some 50 ink-filled firing chambers, each attached to a nozzle smaller than a human hair.

3. An electrical pulse flows through thin resistors at the bottom of all the chambers of all the colors that the printer uses to form a small section of a character or picture on paper.

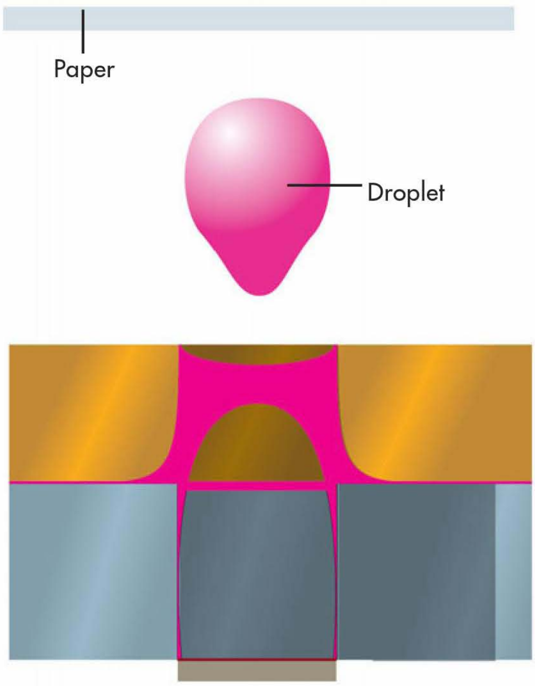




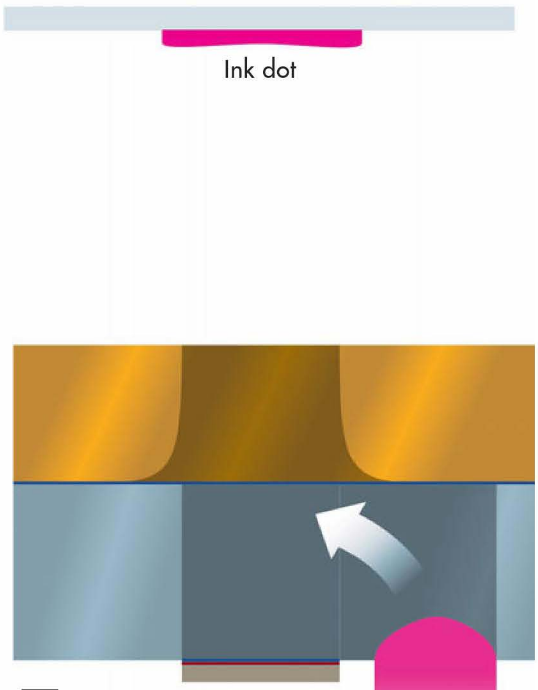

**4** When an electrical current flows through a resistor, the resistor heats a thin layer of ink at the bottom of the chamber to more than 900 degrees Fahrenheit for several millionths of a second. The ink boils and forms a bubble of vapor.



**5** As the vapor bubble expands, it pushes ink through the nozzle to form a droplet at the tip of the nozzle.



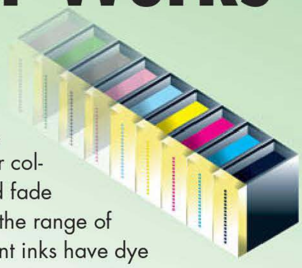

**6** The droplet overcomes the surface tension of the ink, and the pressure of the vapor bubble forces the droplet onto the paper. The volume of the ejected ink is about one millionth that of a drop of water from an eyedropper. A typical character is formed by an array of these drops 20 across and 20 high.



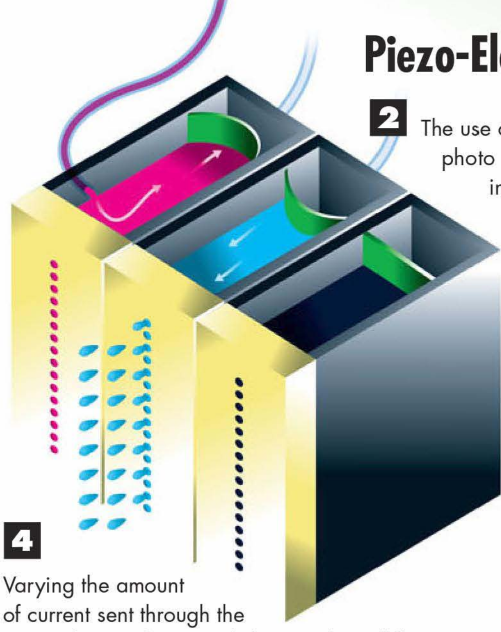

**7** As the resistor cools, the bubble collapses. The resulting suction pulls fresh ink from the attached reservoir into the firing chamber.

# How a Photo Printer Works

## Pigment Inks



1 Photo-quality printers often use more colors of ink—the usual black, cyan, magenta, and yellow, with the additions of light red, light blue, and gray or green. Photo printers use inks that get their colors from either **dyes** or **pigments**. Dye inks are slow to dry and fade when exposed to light and pollutants, but have a wide **gamut**—the range of color shades they create. Pigment inks are dye-based inks. Pigment inks have dye ink's brilliance, but withstand better the assaults of light and the environment.



## Piezo-Electric Nozzles

2 The use of thousands of **piezo-electric nozzles** gives photo printers greater control over the size of sprayed ink drops compared to ink jet nozzles. The back wall of each nozzle is made of **piezo**, a crystalline substance that bends when electricity passes through it. The wall's bending sucks new ink into the nozzle.

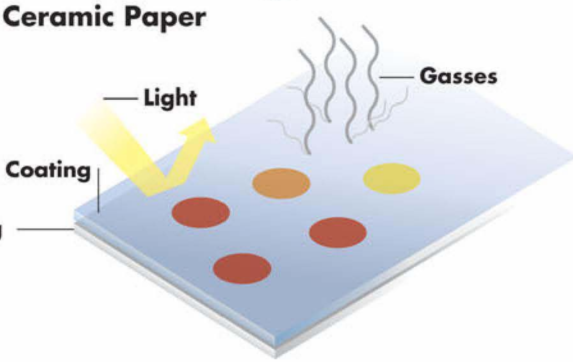
3 Current to the nozzle shuts off. The piezo flexes back into its normal position, forcing a droplet of ink out of the nozzle.

4 Varying the amount of current sent through the piezo changes how much the nozzle wall flexes, which determines the amount of ink pulled into and shot out of the nozzle. The nozzles create drops as small as two picoliters.


Ceramic Paper

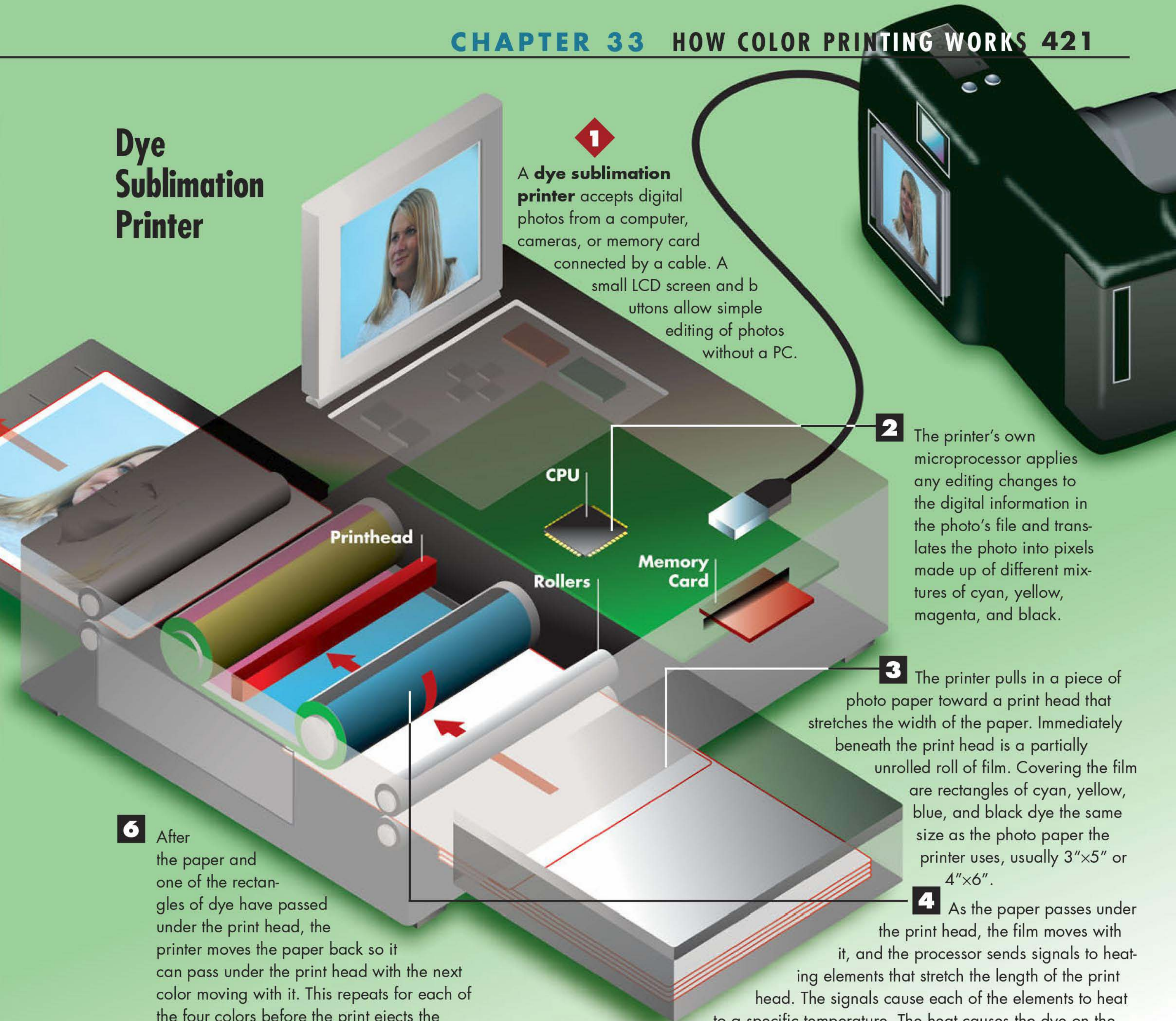
## Photo Paper

1 For the highest quality and longest lasting images, photo printers use special papers made to work with the type of ink the printer uses. **Ceramic-coated porous papers** absorb inks quickly, but their ceramic coating leaves dyes exposed to light and gases. **Plastic-coated swellable papers** protectively encapsulate particles of dye and pigment when the ink seeps into the paper's fibers.




Swellable Polymer Paper

# Dye Sublimation Printer

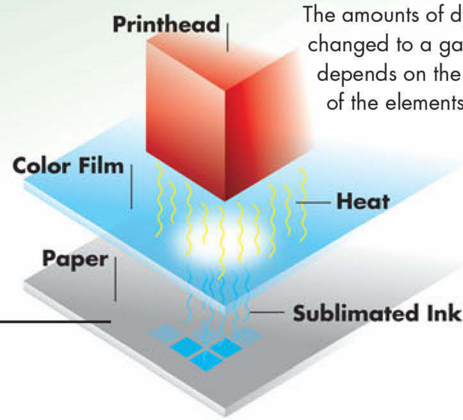



**1** A **dye sublimation printer** accepts digital photos from a computer, cameras, or memory card connected by a cable. A small LCD screen and b uttons allow simple editing of photos without a PC.

**2** The printer's own microprocessor applies any editing changes to the digital information in the photo's file and translates the photo into pixels made up of different mixtures of cyan, yellow, magenta, and black.

**3** The printer pulls in a piece of photo paper toward a print head that stretches the width of the paper. Immediately beneath the print head is a partially unrolled roll of film. Covering the film are rectangles of cyan, yellow, blue, and black dye the same size as the photo paper the printer uses, usually 3″×5″ or 4″×6″.

**4** As the paper passes under the print head, the film moves with it, and the processor sends signals to heating elements that stretch the length of the print head. The signals cause each of the elements to heat to a specific temperature. The heat causes the dye on the film to **sublimate**—change directly from a solid to a gas. The amounts of dye changed to a gas depends on the heat of the elements.




**5** When the clouds of dye meet the paper's surface, they immediately cool to solids again.

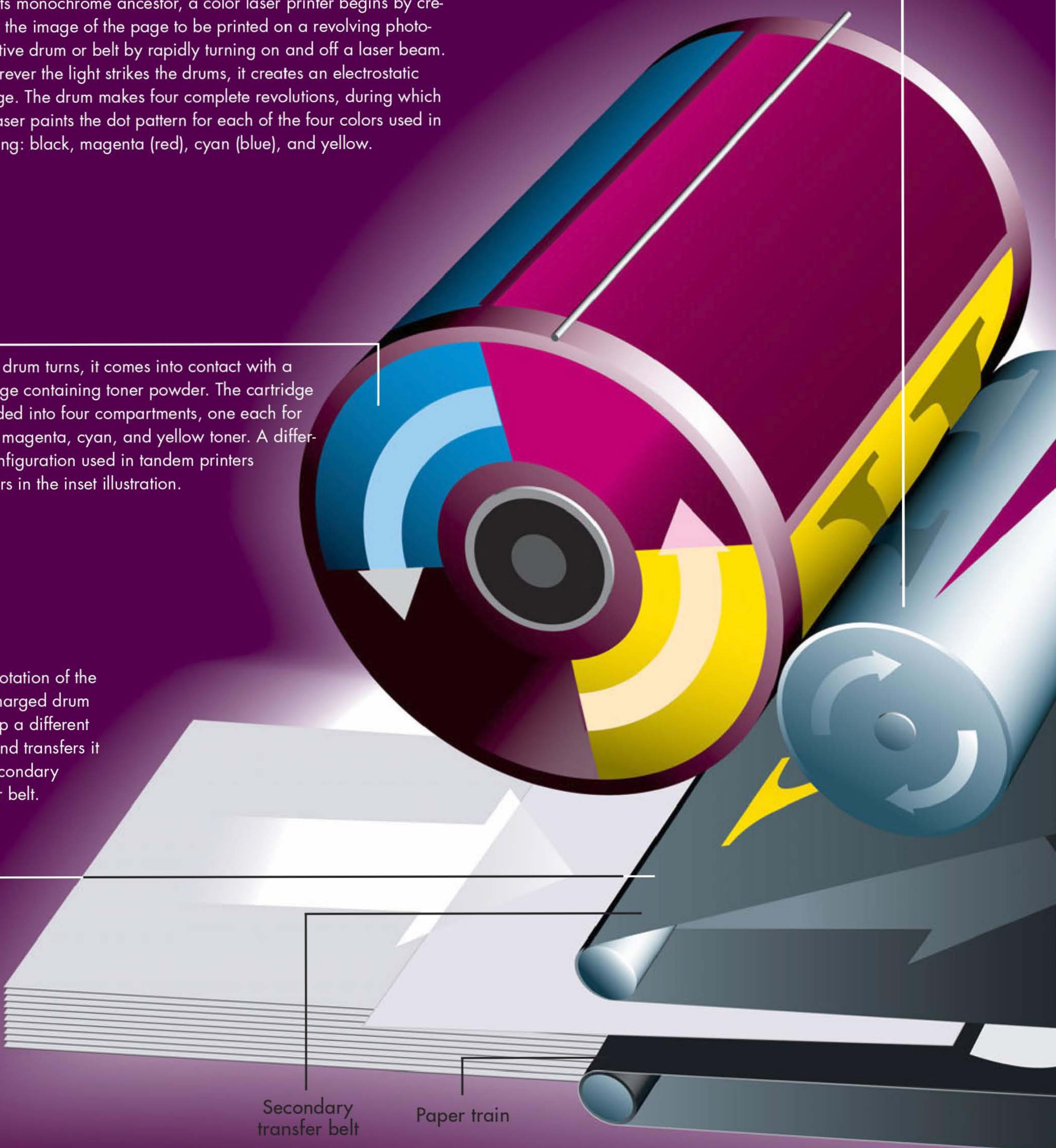
**6** After the paper and one of the rectangles of dye have passed under the print head, the printer moves the paper back so it can pass under the print head with the next color moving with it. This repeats for each of the four colors before the print ejects the finished print.

# How a Color Laser Printer Works

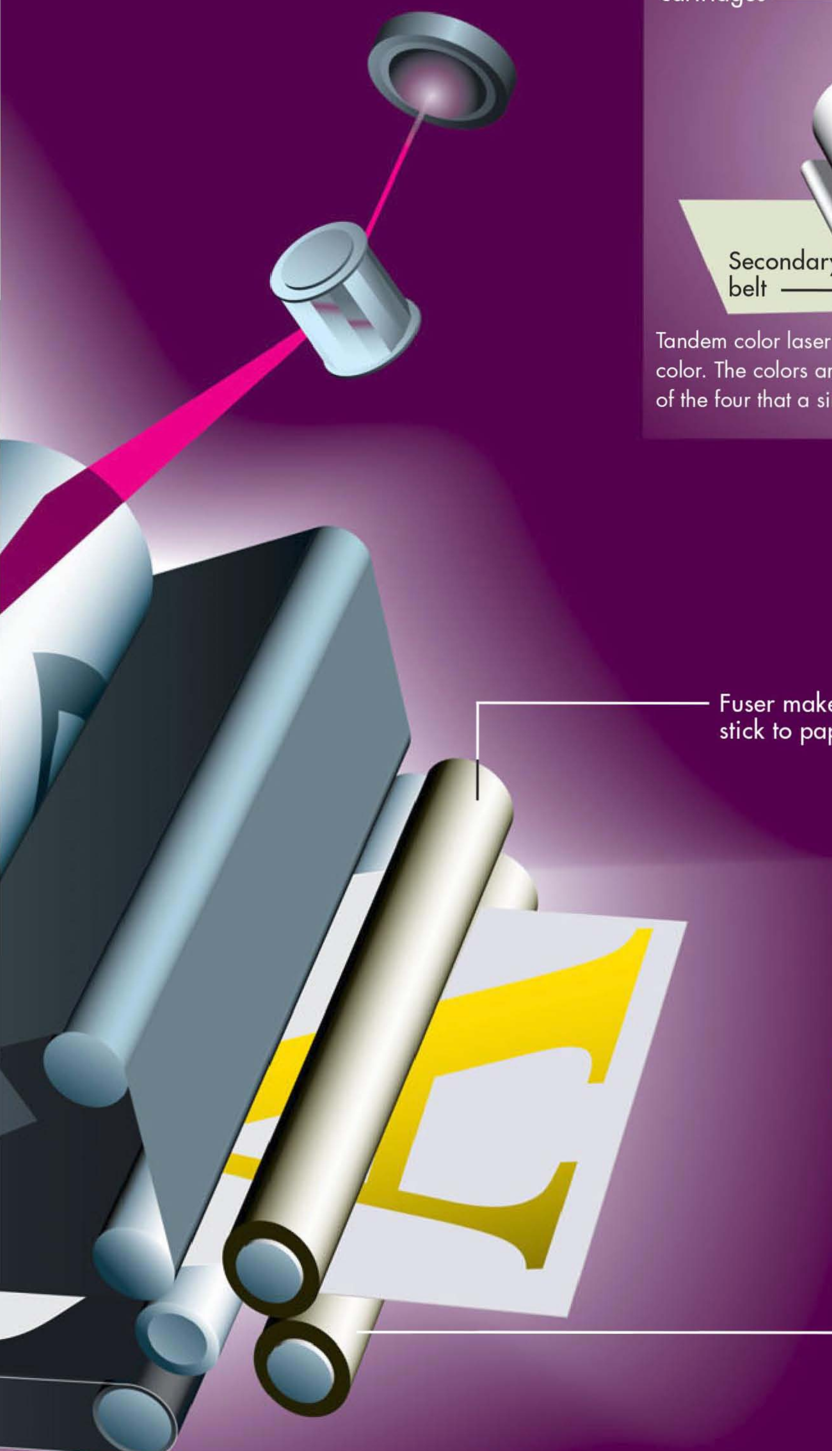
**1** Like its monochrome ancestor, a color laser printer begins by creating the image of the page to be printed on a revolving photosensitive drum or belt by rapidly turning on and off a laser beam. Wherever the light strikes the drums, it creates an electrostatic charge. The drum makes four complete revolutions, during which the laser paints the dot pattern for each of the four colors used in printing: black, magenta (red), cyan (blue), and yellow.

**2** As the drum turns, it comes into contact with a cartridge containing toner powder. The cartridge is divided into four compartments, one each for black, magenta, cyan, and yellow toner. A different configuration used in tandem printers appears in the inset illustration.

**3** Every rotation of the laser-charged drum picks up a different color and transfers it to a secondary transfer belt.

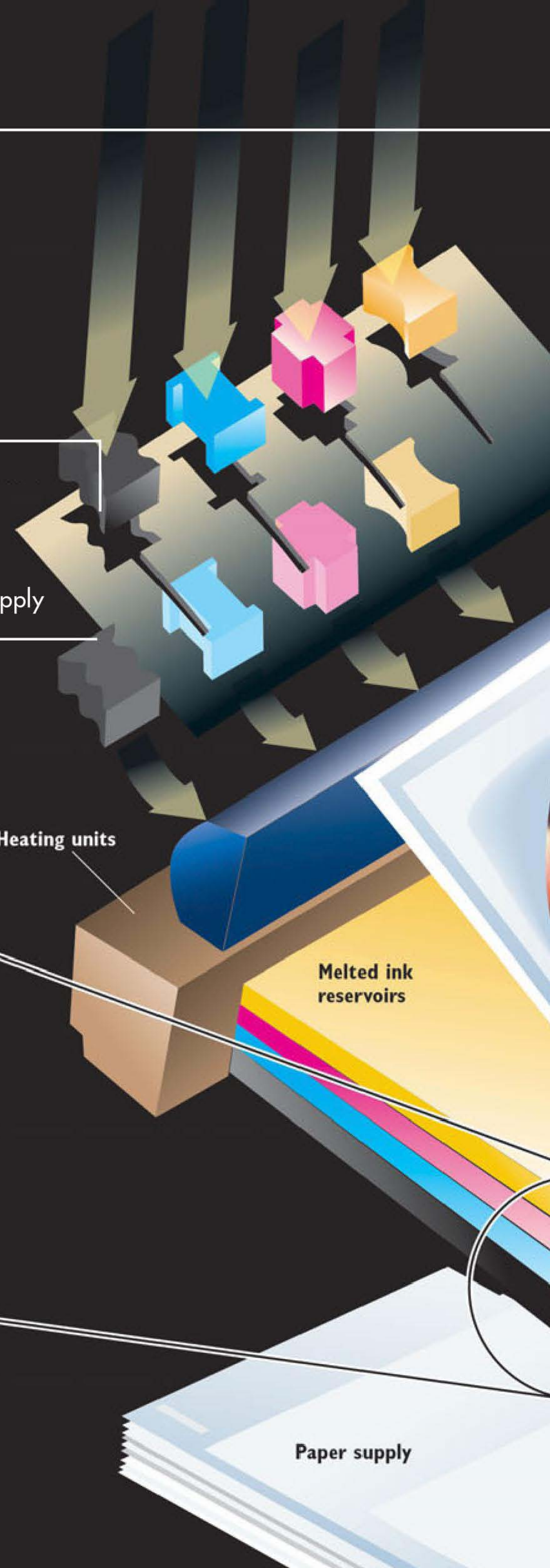



Tandem color laser printers have separate lasers and photostatic drums for each color. The colors are transferred to the secondary belt in one turn of the belt instead of the four that a single-drum color laser printers require.




**4** After all the colors are on the belt, the paper train grabs a sheet of paper from a bin and moves it along a path beneath the secondary belt. A roller presses the paper against the belt, transplanting all the color toners to the paper at one time.

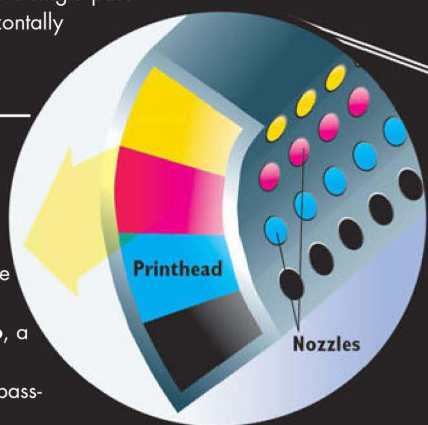
# How a Solid-Ink Color Printer Works



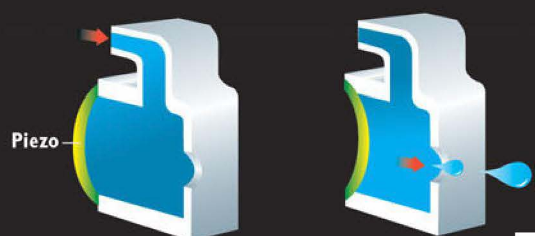

**1** A solid-ink color printer such as the Tektronix Phaser 350, which prints six pages a minute on ordinary paper, uses inks that are solid at room temperatures. They are loaded into the printer as wax-like blocks for each of the four colors the printer uses. The holes for loading the blocks are shaped differently for each color to prevent loading a color into the wrong hole.

**2** As each color is needed, four heating units melt the inks into liquids that collect in separate reservoirs in the print head.

**3** Ink flows to a print head that, in the Phaser 350, consists of 88 vertical columns of four nozzles, one for each color. The print head extends across the entire width of the page, and it produces an entire line of dots in a single pass while moving back and forth horizontally only 1/2 of one inch.

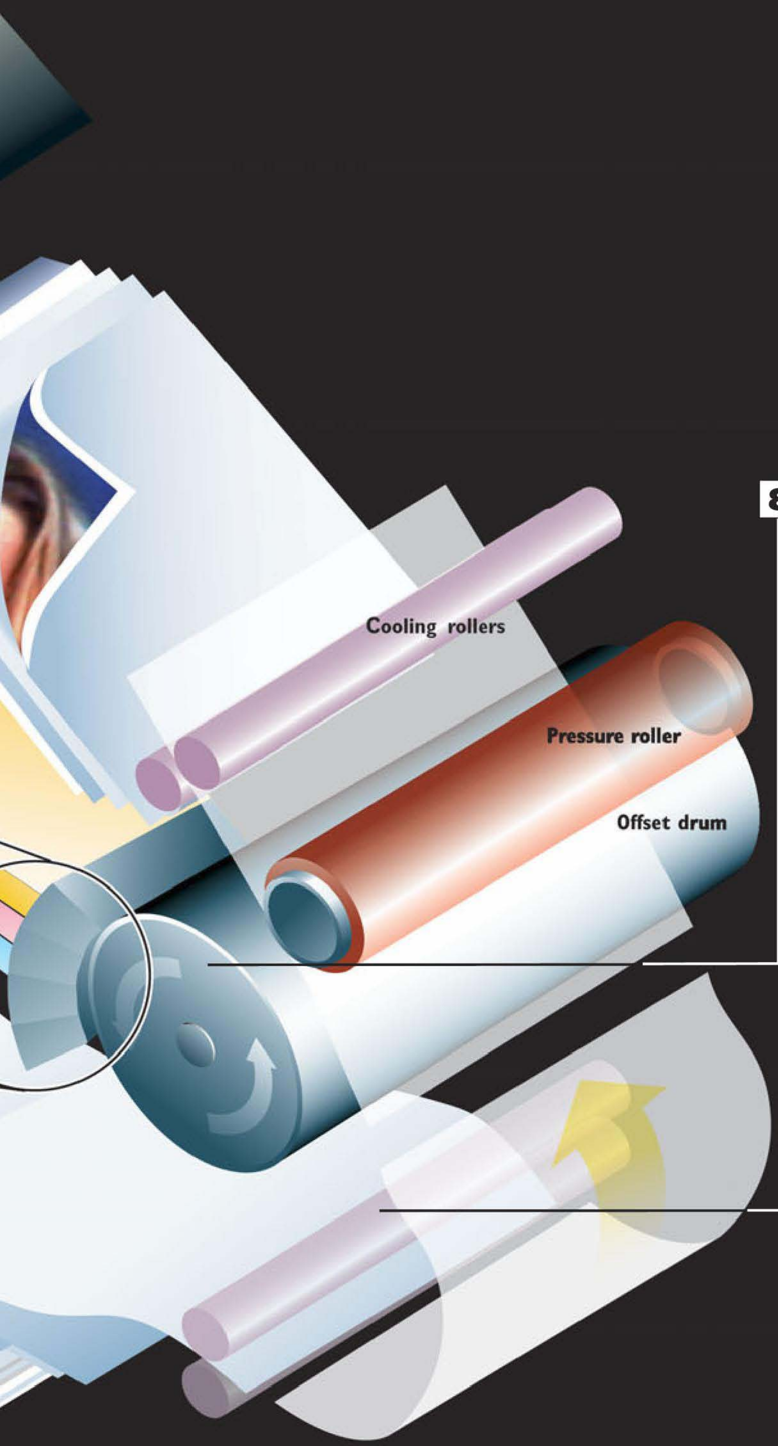



**4** Each nozzle is operated by a **piezo controller**. The controller varies the amount of electricity passing through the rear wall of the individual ink chambers for each nozzle. The wall is made of **piezo**, a crystalline substance that flexes in response to the amount of current passing through it.




**5** As the wall moves back, it sucks ink in from the reservoir and back from the mouth of the nozzle. The farther the wall flexes, the more ink it pulls in, allowing the controller to change the amount of ink in a single dot.

**6** When current changes, the wall flexes in, vigorously pushing out a drop of ink onto an offset drum coated with a silicone oil. The drum is slightly warm to keep the ink in a liquid state.




**8** As the offset drum turns, the ink comes into contact with the paper, which picks up the dots of color. The pressure roller pushes the ink into the paper to prevent it from spreading on the paper's surface, which would blur the image. The ink dries and refreezes immediately as it passes between two unheated rollers. The drum revolves 28 times to transfer enough ink to cover an entire sheet of paper.

**7** The paper supply feeds a sheet of paper into a path that takes it between the offset drum and a roller pressing against the drum.

# Index

## NUMBERS



## A

## B

## C

## E

## J - K



## L

# N

## O



## P

# Q - R

## S

## U



## V

## W

# X - Y - Z